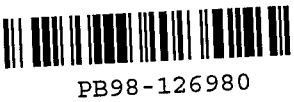
Final Report
TNW 98-02



TRANSIT TIME INTERNET ACCESS

by
Kenneth J. Dueker
Janet Vorvick
Ray Jackson
Jun Qiu

Center for Urban Studies
College of Urban and Public Affairs
Portland State University
Portland, Oregon 97207-0751

Transportation Northwest
(TransNow)
Department of Civil Engineering
135 More Hall
University of Washington, Box 352700
Seattle, WA 98195-2700

January 1998

TECHNICAL REPORT STANDARD TITLE PAGE

1. REPORT NO. WA-RD ___._TNW 98-02	2. GOVERNMENT ACCESSION NO.	3. RECIPIENT'S CATALOG NO.
4. TITLE AND SUBTITLE TRANSIT TIME INTERNET ACCESS		5.REPORT DATE January 1998
		6. PERFORMING ORGANIZATION CODE
7. AUTHOR(S) Kenneth J. Dueker, Janet Vorvick, Ray Jackson, and Jun Qiu		8. PERFORMING ORGANIZATION REPORT NO. TNW 98-02
9. PERFORMING ORGANIZATION NAME AND ADDRESS Transportation Northwest Regional Center X (TransNow) Box 352700, 123 More Hall University of Washington Seattle, Washington 98195-2700		10. WORK UNIT NO.
		11. CONTRACT OR GRANT NO. DTRS95-G-0010
12. SPONSORING AGENCY NAME AND ADDRESS Tri-Met 4012 SE 17th Avenue Portland, Oregon 97202		13. TYPE OF REPORT AND PERIOD COVERED Final Report
		14. SPONSORING AGENCY CODE
15. SUPPLEMENTARY NOTES This study was conducted in cooperation with Portland State University.		
16. Abstract Transit Time Internet Access (TTIA) is a Web (World Wide Web, WWW) prototype application which delivers real-time bus schedule information to users of the Internet. TTIA allows a bus rider to request and receive schedule deviation information about a specific bus at a specific timepoint. TTIA is part of a larger project whose goal is to evaluate the effect on riders; behavior and riders' level of satisfaction of actual arrival time information. TTIA is accessible at http://www.upa.pdx.edu/TTIA/		
17. KEY WORDS real-time, schedule adherence, World Wide Web, Automatic Vehicle Location, Advanced Public Transportation Systems		18. DISTRIBUTION STATEMENT No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22616

19. SECURITY CLASSIF. (of this report) None	20. SECURITY CLASSIF. (of this page) None	21. NO. OF PAGES 48	22. PRICE $6.50

Table of Contents

Table of Figures

Preface

Kenneth J. Dueker, the director of the Center for Urban Studies at Portland State University, is the principal investigator of this project. Janet Vorvick, a research assistant at the Center for Urban Studies, was responsible for the implementation of the main module of the Transit Time Internet Access (TTIA) program. Jun Qiu and Ray Jackson, graduate research assistants, implemented the Interpolated Time Module for TTIA and the client-side image maps, respectively. Jun Qiu is the author of Appendix A, *The Interpolated Time Module for TTIA,* and Ray Jackson is the author of Appendix B, *Feasibility of Placing Bus Route Maps on the Internet.*

Disclaimer

Abstract

Transit Time Internet Access (TTIA) is a Web (World Wide Web, WWW) prototype application which delivers real-time bus schedule information to users of the Internet. TTIA allows a bus rider to request and receive schedule deviation information about a specific bus at a specific timepoint. TTIA is part of a larger project whose goal is to evaluate the effect on riders' behavior and riders' level of satisfaction of actual arrival time information.

TTIA is accessible at http://www.upa.pdx.edu/TTIA/

Keywords

real-time, schedule adherence, World Wide Web, Automatic Vehicle Location, Advanced Public Transportation Systems

1. Introduction

Transit Time Internet Access (TTIA) is a prototype of a Web application that allows bus riders to receive information about a bus they intend to ride. Through the TTIA Web pages, a user of the bus system of Tri-Met, the Tri-County Metropolitan Transportation District of Oregon, can find out if their bus is expected to come to their bus stop early, late or on schedule. TTIA is an exploratory research project to examine innovative ways of delivering bus schedules and actual arrival times to customers. The ability to deliver actual arrival time information has been made possible by Tri-Met's implementation of a new bus dispatch system (BDS). The BDS package includes automatic vehicle location(AVL) technology that uses a global positioning system (GPS) to track the location of buses. Bus locations are then related to digital representations of bus routes and schedules.

Currently, users can access schedule information by phone or on the Internet. Tri-Met is installing a new telephone information system that will deliver actual arrival time information as well as schedule information. In fact, the TTIA system taps into the stream of exception messages sent by the BDS for the new telephone system, and processes it for delivery via the Internet.

TTIA is part of a larger project evaluating uses of data coming from the new BDS. This larger purpose addresses the implications of improvements in the quality of transit service. In this context we are interested in the manner in which better arrival time information will change users' travel behavior. Will it increase transit patronage or reduce the erosion of ridership over time? Similarly, can analysis of on-time performance and bus bunching lead to strategies and procedures for dispatcher interventions that can improve on-time performance? Will better information reduce bus bunching, and will those improvements lead to increased patronage and reduction of ridership losses over time?

A user of TTIA receives an actual arrival-time estimate by opening an html[1] document. This corresponds to using a Web browser to click on a link on a Web page. Assisting the user, TTIA offers a series of four choices that determine which bus is the user's bus-of interest and at what location they plan to board. The schedule update page for the user's bus-of-interest reports either that the bus is early, is late, is running on time or that there is no information available for the bus-of-interest. Proposed extensions of TTIA would report reroute information, (snow routes, for example) and also alert the user if the bus of-interest is out of operation.

TTIA is intended for the use of regular bus riders. Therefore some aspects of the bus system don't need to be explained in the process of guiding the user through the Web pages that offer choices of routes, etc. The assumption is made that TTIA users know where they are, know where they want to go, can read Tri-Met's printed bus schedules and can navigate the Web. Users who haven't experienced the Web have to learn only the

[1] The Hypertext Mark-up Language is used to encode the instructions to a Web browser that allow Web pages to be displayed with links to other pages, italics, different sized fonts and other features.

actions that any Web user learns: point and click to open a page and bookmark those pages used often.

The top page of TTIA offers a choice of routes. The next page offers a choice between two bus directions which usually correspond to inbound and outbound. The third choice the user must make is the choice of a bus stop. The choices offered correspond to the information provided on a paper schedule - ten or twelve timepoints along the route are listed - and help in the form of maps or more detailed stop lists is offered. After selecting a location, the user is presented with a list of scheduled bus arrival times at that location for the period of time that begins fifteen minutes before the current time and ends 45 minutes after the current time. The user then clicks on the arrival time of the bus they plan to take causing the execution of a program that searches a database of messages to discern whether there is a reported deviation for that particular bus. The information from the appropriate message is relayed to the bus rider in a Web page.

One of the greatest challenges for TTIA is the interpretation of the messages in the database. The messages used by TTIA are schedule exception messages that are automatically sent by radio from a bus that is off schedule to the Bus Dispatch System. The absence of a message for a certain bus usually means that the bus is on schedule. However, it can have other meanings, as we will discuss later. A message indicating that the bus was reported to be late may mean that the bus will be late to the user's stop, or it may not, since the bus may make up lost time. TTIA presents the real-time data and interprets it as best it can. The user is presented with a disclaimer explaining the limitations of the system so that they understand what the information means - and what it doesn't mean.

TTIA is a prototype rather than a commercial system. The construction of a commercial system precludes the flexibility and speed we desired for our development process. Specifically, the development process we employed is called *rapid prototyping*[2]. Rapid prototyping is a technique that addresses the problem often encountered at the end of software projects that the final product is a program that no one wants. Eliciting information from the end user about the program they want is so difficult that discussion has come to be seen as inadequate. To facilitate the expression of the requirements for a program, a prototype is created and shown to the user. However, no attempt is made to give the prototype full functionality. For example, a first prototype for TTIA had a reasonably pretty user interface, but had only one active link on each page. For the user to experience the functioning of TTIA, he or she was required to pick the route and stop that the programmer had chosen.

In addition to the absence of full functionality, a program that is part of the rapid prototyping process of software engineering often is lacks the organization and structure of a production program. The purpose of the prototype is to facilitate the specification of the program to be made, so it isn't important to make it tidy. Even the choice of languages and software tools can be different for the prototype and the commercial

[2] More information about rapid prototyping is available in any up-to-date software engineering book. For example, On Time, Within Budget, E. M. Bennatan, John Wiley and Sons, 1995.

system. It's not unusual for the system designer's idea of the program and the user's to be substantially different even after the initial discussions of the problem to be solved. So it would be surprising if the programmer happened to choose the structure and language for the prototype that turned out to be the best for the commercial system. The implication is that one does not improve the prototype to create the commercial system. One discards the prototype and begins the implementation taking advantage of all the information learned in the rapid prototyping process.

As a consequence of the use of rapid prototyping, it was not desirable to incorporate every Tri-Met bus route into the TTIA system. To reduce the scope of the project we used the weekday schedules for seven routes. Also because of the limitations of a small scale prototype, some attractive features of TTIA are only partly implemented. For example, a program to help a bus rider locate their stop if it is not a timepoint is not complete. And the information TTIA delivers is drawn from a static database of schedule exception messages. Acquiring the schedule exception messages in real-time was hampered because communication between TTIA and the Tri-Met computer that receives the exception reports has been problematic. Still, the prototype is adequate for assessing the usefulness of a real-time information delivery system. The experience gained from creating the system allows us to suggest some design and implementation pitfalls to avoid.

The remainder of this report shows the TTIA Web pages and explains the design of TTIA, the specifics of the implementation, the issues of scope and the problems we encountered. Finally we offer some advice for those creating a TTIA-like system or any system to deliver real-time transit information.

2. Literature Review

TTIA is designed to deliver real-time information to a transit user. Similar systems are in place serving the riders of several transit systems in the U. S. A larger number are planned or under development. The following section looks at some of the real-time systems for transit users.

Systems that deliver real-time information are generally classified as pre-trip or en-route traveler information systems. TTIA is designed to be a pre-trip information system, but differs from most pre-trip systems in its delivery of real-time information. Most pre-trip systems are designed to help a transit user plan a trip. Delivery of the information one would find on a printed schedule is almost universal, but a real-time component is rare.

Among the pre-trip systems that do share TTIA's goal of providing real-time information is the Busview[1][2] program developed by the University of Washington and the King County Department of Metropolitan Services. This program shows, on a map, the location of busses on a TIGER representation of a street network. Busview is accessible on the Web from certain computers on the campus of the University of Washington. Like TTIA, Busview allows users to plan their walk to the bus stop based on real-time information in addition to fixed-schedule information. Busview uses signpost-based Automatic Vehicle Location (AVL) to locate busses. Users query a bus icon to obtain

schedule information, but must personally interpolate from the current location of the bus to estimate an arrival time at their bus stop.

Another pre-trip system for delivering real-time information to bus riders is operating in Minneapolis. Two methods of delivering the information are in use by this system called Travlink[3][4]. First, Videotext terminals have been distributed to a group of people recruited for the project. Data collected indicates that this method of accessing real-time bus stop information is attractive. Users are wanting more information. Second, three kiosks provide real-time information. At the kiosks, real-time information is not used much. They are used mostly for trip planning.

Several other pre-trip systems were in the planning stages as of January 1996. Kiosks that offer trip planning augmented with real-time information are part of the transit plans in Corpus Christi[3], Cincinnati[3][5], and Atlanta[3][6]. In Ann Arbor, Michigan, plans include a demonstration of wayside signs that give the amount of time until the next bus's arrival. Cable TV is also among the media being considered in Ann Arbor[3].

The focus of pre-trip information, in the past, has been trip planning and the dissemination of schedule information. En-route information is meant to facilitate transfer decisions and itinerary modifications. However, it's not clear whether a kiosk at a bus stop is providing pre-trip information or en-route information. Though TTIA is clearly a pre-trip system (the user is seated in front of their computer), it has a lot in common with some systems that are discussed in the context of en-route systems. If a kiosk at a bus stop is an enroute system, then TTIA could be converted to an en-route system simply by locating an appropriately connected computer in a kiosk at a bus stop. Twenty-one en-route systems in North America were identified in the U.S. Department of Transportation's 1995 report, Review and Assessment of En-Route Transit Information Systems[7]. A few of these use on-board signs to display real-time information, and are, thus, very different from TTIA, but most use kiosks.

Whether classified as pre-trip or en-route information services, these many systems reflect the current enthusiasm for making use of the large amount of data now available because of the installation of AVL systems. Clearly riders can benefit from access to the information and transit providers are eager to make it available to them.

3. The Look and Feel of TTIA

The starting point of the TTIA system is a top page that has three sections (three frames). Though all three are visible at once on a browser, they are shown here in succession, as if they were three Web pages. The first gives general information about TTIA. The second offers links to other parts of TTIA, to an information page and to Tri-Met. And the third is the page requesting the initial response from the user — their choice of routes.

Updates for Bus Travel

Tell us which bus you ride, and we'll tell you whether your bus is running on time today. If you haven't already indicated your bus route, you'll want to begin with the first step.

This is a project of the Center for Urban Studies at Portland State University in cooperation with Tri-Met.

Figure 1: Top Page, Part One

- Weekdays
- Saturdays
- Sun/Holidays

The List of Routes

Tri-Met

How does this program work?

Figure 2: Top Page, Part Two

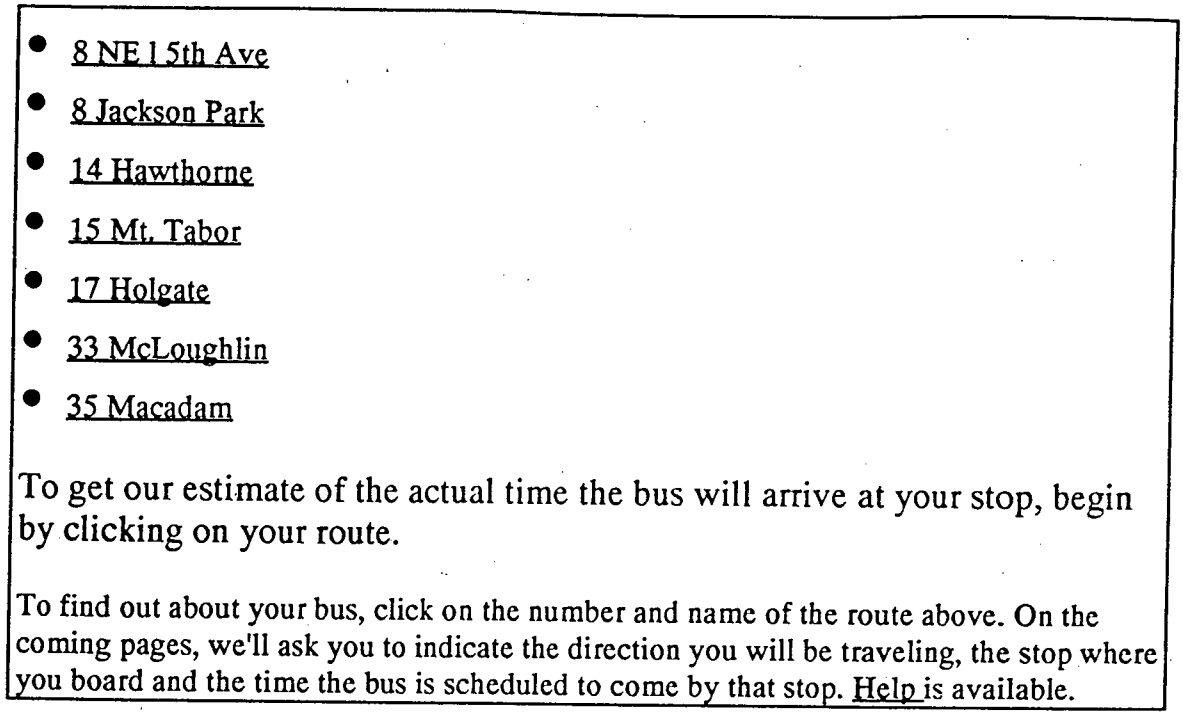


Figure 3: Top Page, Route Selection

When the user has chosen a route, a page asking them to indicate their direction of travel is displayed. For example, if the user chooses route 15, the page below appears.

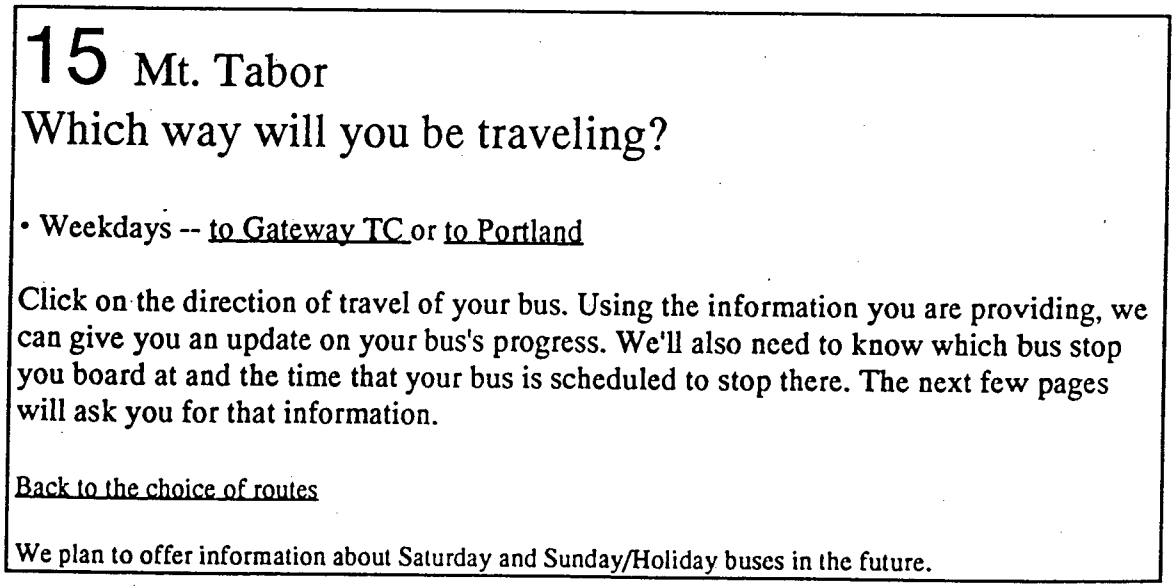


Figure 4: Direction of Travel Choice

When the route and direction of travel have been chosen, the user sees a Web page asking them to identify their bus stop. If they can identify a nearby time point, then they access

the next page by clicking on that time point. If the user needs help identifying their bus stop, they can use either the map accessible through the hot-linked word *map,* or ask for help by clicking on the hot-linked word *help.*

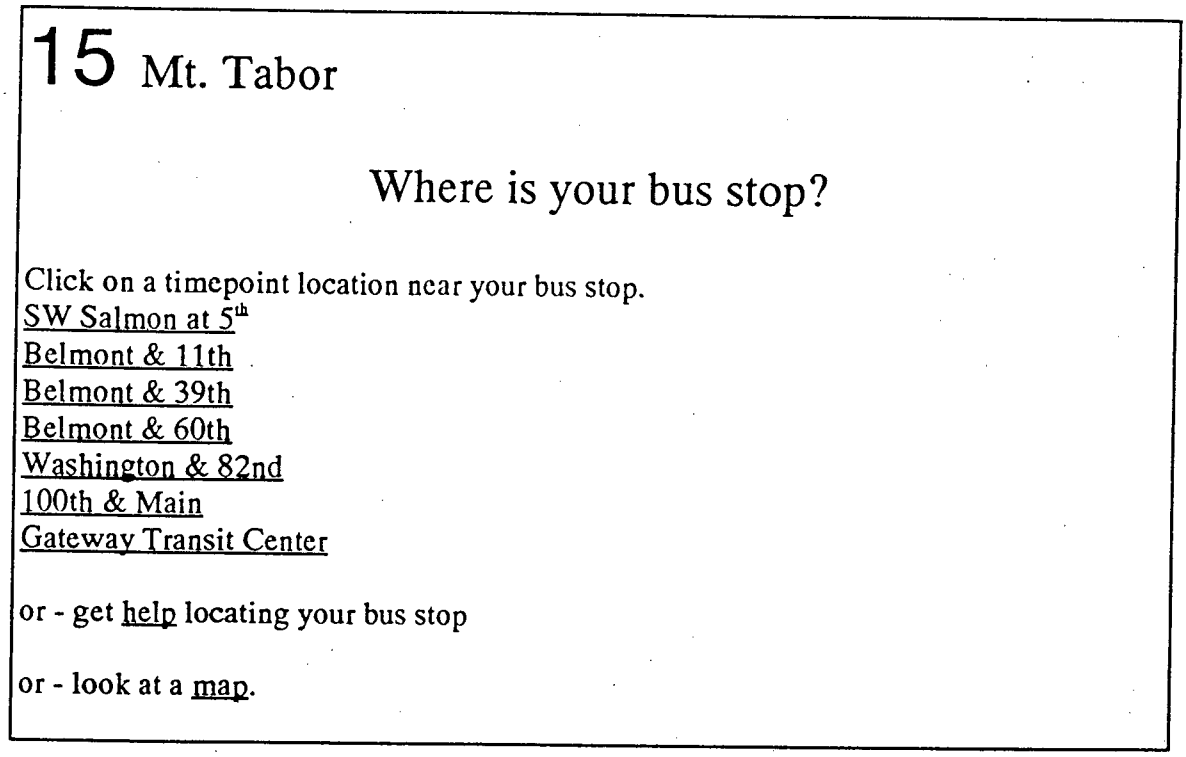


Figure 5: Bus Stop Selection

Now that the stop is determined, TTIA shows a page offering a choice of times-of-day. In the interest of keeping the list short, the offered times-of-day are not all the times that the bus is scheduled to arrive at the stop in one day. The list of all the times can be very long. Consequently, we assume that the user is making the request for update information when the current time and the time of their bus are not much different. If the short list of times isn't adequate, they can request a larger list.

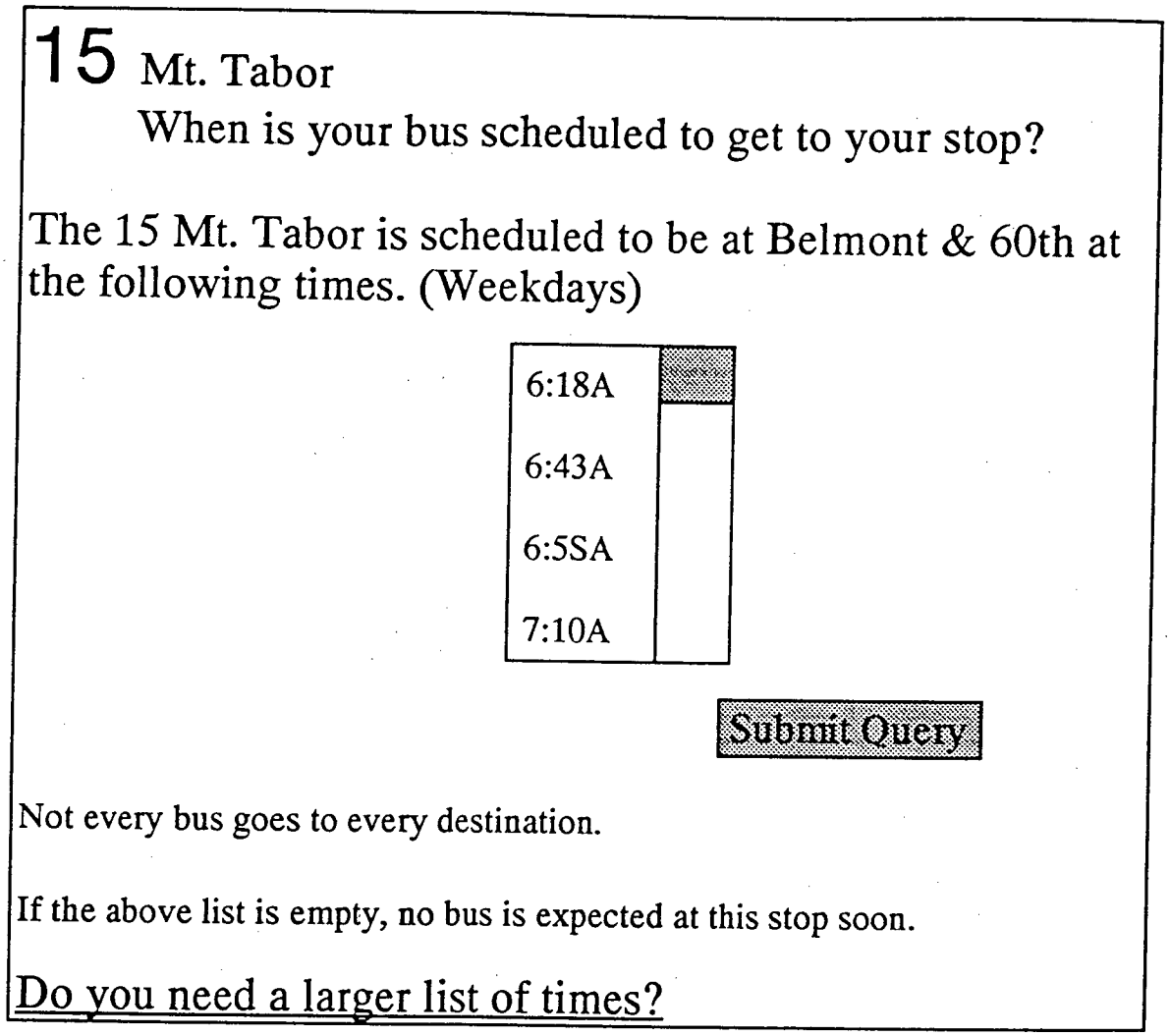


Figure 6: Time Selection

There are four responses TTIA might give a user after the selection of the specific bus they are planning to catch. One message says that the bus appears to be running on time. The next reports some schedule deviation. A third alerts the user that the bus has already passed the stop. Finally, if the request is made too early, TTIA advises the user accordingly.

Each of the four messages contains advice about interpreting the information delivered, bookmarking Web pages and reloading pages. Each presents links to parts of the TTIA system (for asking about another bus, for example) and links to Tri-Met's Web services. Those parts of the Web pages common to all four have been reproduced here only for the first page.

Update Information for the 15 Mt. Tabor:

The scheduled time for the **15 Mt. Tabor** is **7:44A** at **SW Salmon at 5th** (timepoint location).

SW Salmon at 5th . . . 7:44 AM

This page was assembled at 7:32.	Reload

This bus has not been reported late. However, its status could change. Changes in traffic, ridership and weather may cause the bus to run either early or late. When snow and ice are on the roads, the information available from the Customer Services (238-RIDE) is more accurate than this page. This is the best information for the bus at this time.

Since bus information changes frequently, check the time this page was assembled. For the latest information, use your browser's reload button, or reload from this link. Sometimes a browser will save a copy of a Web page, so it's important to reload for the latest information.

Bookmark this URL. If you want to ask about this bus again (perhaps later today, or tomorrow) you can use the bookmark facility of your browser to save this page. The next time you open this page, the most recent information will be displayed.

- Back to the list of times for the 15 Mt. Tabor at SW Salmon at 5th
- Back to the top page.
- Tri-Met's Customer Service and Personalized Trip Planning
- Tri-Met's Snow Routes and Detours Page
- Tri-Met's Bus Information Page
- Tri-Met's Bus Schedules

Figure 7: TTIA On-Time Page

SW Salmon at 5th . . . 7:42 AM

This bus has been reported off-schedule. However, its status could change. Changes in traffic, ridership and weather may cause the bus to be back on schedule soon. This is the best information for the bus at this time.

Figure 8: TTIA Off-Schedule Page

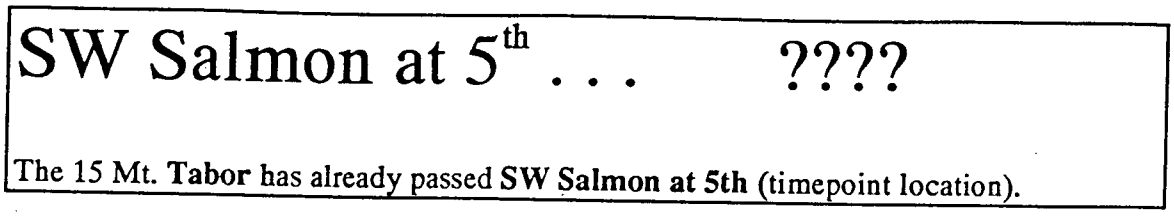


Figure 9: TTIA Bus Passed Page

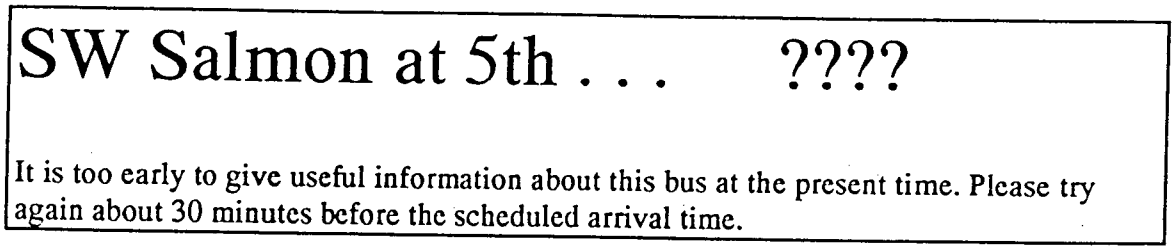


Figure 10: TTIA Too Soon Page

4. TTIA Design

There are basically two parts of the TTIA system. One part collects information from the user and the other delivers information to the user. Both of these parts use a Web browser for their user interface.

Collecting information from the user involves some data about the bus system. For example, the user chooses among the bus routes as they indicate which bus they are interested in. Thus the information collecting part of TTIA presents the user with choices and records the choice. There are four choices to be made:

- What route?
- Direction of travel? Inbound or outbound?
- Which timepoint location?
- What time?

For example, if the answers were route 8, to Portland, SW 6th and Main and 10:15, it would mean that the user plans to board bus 8, inbound, at SW 6th and main at 10:15 and wants to know if it's running late.

A design decision had to be made between creating Web pages to offer the appropriate choices as the user proceeds and creating all of the Web pages ahead of time and simply choosing the right page to serve as the user proceeds. The first approach is dynamic in the sense that pages are made only as they are needed. The second approach is static because the pages are created just once and then they sit around until they are needed.

Since the price of storage space on a computer is small, creating all the pages just once is reasonable. But TTIA with its seven bus routes requires 160 Web pages to offer all the options of route, direction, stop and time. Creating Web pages "on the fly" reduces the amount of space needed, but requires calculations every time a user clicks on a page. The

design decision for TTIA was in favor of static pages partly because static pages were already being created by Tri-Met to provide bus schedules on the Web.

The final step in the information-gathering part of TTIA occurs when the user clicks on a time (the scheduled time the bus comes by their stop). The cgi-bin[3] facility of the Web server (the machine on which the Web pages are located) allows the four pieces of information, route, direction, stop and time, to be communicated to the second part of TTIA, the part which delivers the schedule adherence information. The major work of the second part is the creation of the update page for the user's bus-of-interest.

To create the update page, a program is executed. It receives the user's input and checks the database of schedule exception messages to see if the bus-of-interest is currently known to be off schedule. This database of exception messages is indexed by an identifying number for each bus, not by route, direction, and/or time. So it is necessary to look up the identifier for the bus of interest. For example, the user who plans to board bus 8 headed to Portland at SW 6th and Main at 10:15 wants a specific bus running a trip that Tri-Met refers to as Train 1733. If the database contains an exception message for Train 1733 which has in the time field the number 11, then the bus is eleven minutes late.

Naturally, the organization and referencing of the database are crucial We'll leave these details for the following section on implementation. The design aspect of the update page generation need only assert that the database is consulted and the required information is obtained.

Using the cgi-bin facility again, the output of the program is sent to the user as a Web page. Most of the update page is the same for every query. The parts that change are the estimated time that the bus will arrive at the user's stop, the repetition of the information identifying the bus-of-interest (for the user's benefit) and the disclaimer that accompanies an estimated arrival time.

The picture of information flow through the TTIA system follows.

[3] The Common Gateway Interface is a specification for transferring information between programs. The "bin" extension is short for binaries, which refers to the kind of programs that are usually put in the cgi-bin directory.

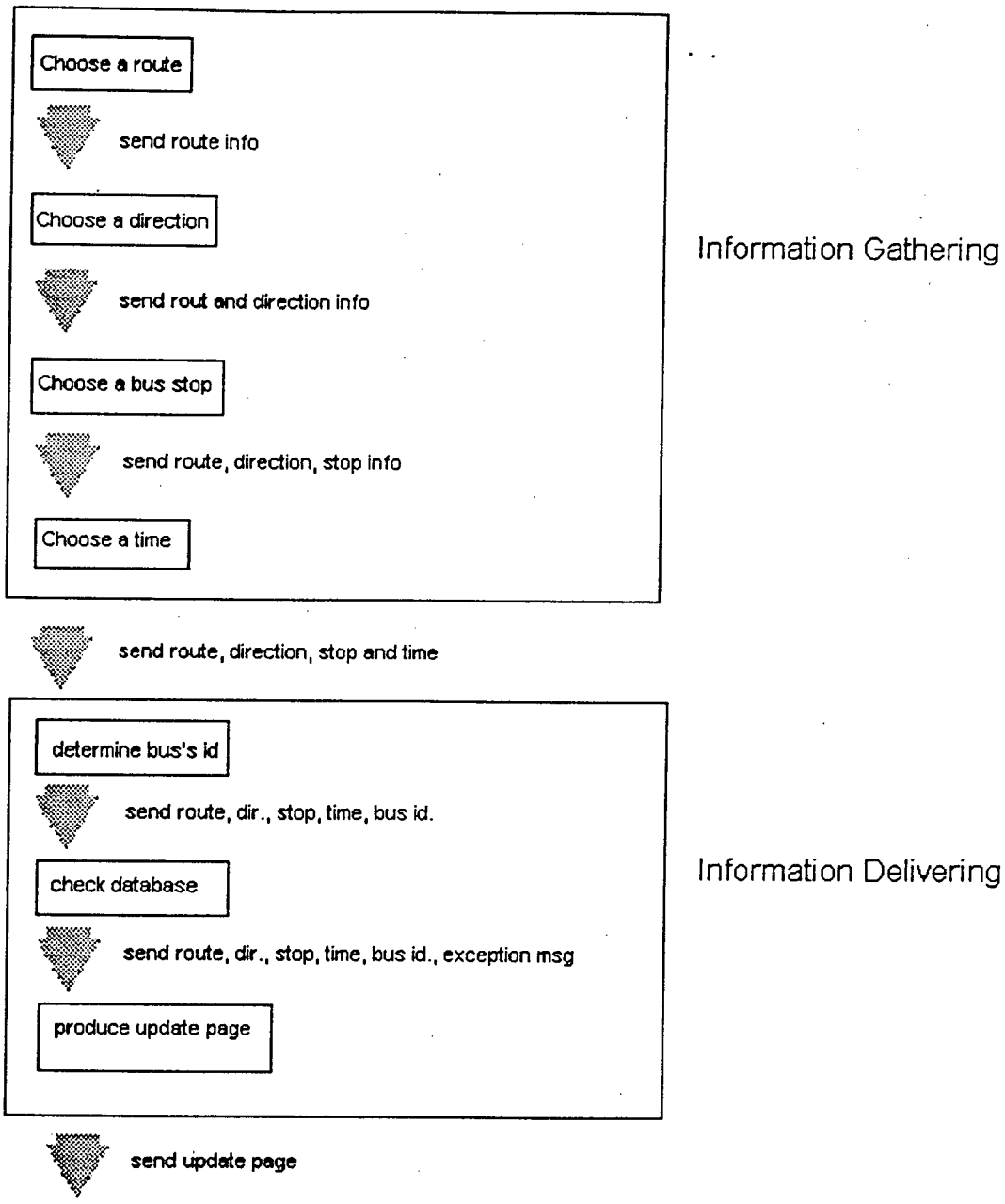


Figure 11: The flow of data through TTIA

5. TTIA Implementation

The TTIA system, as described above, could have been implemented a number of different ways. For example, the program that uses the information provided by the user to determine whether there is a schedule exception message for the bus-of-interest could

have been written in any of several languages. Such a program could pass information to a commercial database program or have its own code for searching. This section explains the choices we made for the details of TTIA.

The 160 Web pages that make up the information gathering part of TTIA form a tree structure. There is one page for the initial choice made by a user: the choice among bus routes. There are seven pages at the second level representing the seven routes included in this prototype. Each offers the choice of inbound or outbound travel. In other words, there is no generic page that solicits from the user their preference for inbound or outbound travel. The seven pages that serve that function are route-specific, though they are similar in format.

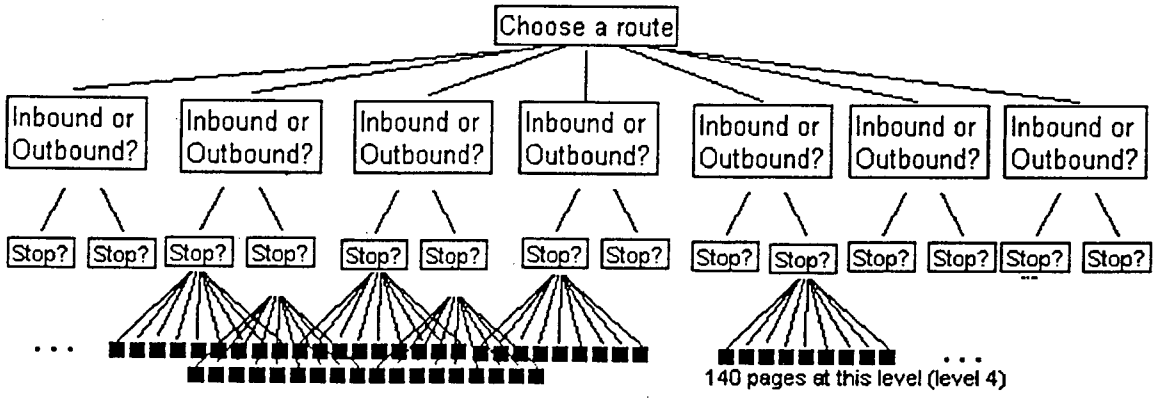


Figure 12: The Tree Structure of TTIA Web Pages

Because there are two choices available to the user at the second level there are 14 pages on the third level Each of these offers about ten choices (the timepoints for the route), requiring about 140 pages at level 4. The 140 pages are the ones offering the user the choice of a specific Train, that is, a bus arriving at their stop at a specific time. Since the act of clicking on one of the times displayed starts a program, this is the last level of static Web pages.

There is also a map file for each route and direction. These maps allow the user to locate their bus stop (or nearby timepoint) by clicking on its location on the map. The number of route times two directions determines the number of static Web pages that are needed to offer this alternative. The benefit is large, however, because some riders are more able to find their stop on a map than to find it by name. Appendix B contains more about these client-side maps.

Another option for the user of TTIA branches from the third level of the tree structure. A program that helps the user locate a bus stop between time points is available. This program presents, via a Web page, a list of all stops for the route. Naturally the time points are among them. The user can scroll through the stops to find their stop. The implementation of this module was more complex than expected, since there is not just one list of stops for a route and direction. Sometimes a bus will be scheduled to serve only

part of a route. Sometimes busses with a single route number alternate between two sets of stops for some part of the route and some part of the day. These details made it necessary to offer several lists of stops to the user. More information about this module, the Interpolated Time Module, is in Appendix A. This module uses the complete set of routes and the Saturdays and Sundays/Holidays information in addition to the Weekdays information we used for the remainder of TTIA. This provided valuable experience with the large set of data for the bus system.

Because the TTIA Web pages are static and arranged in a tree, the hypertext links on the fourth level can contain all the information needed by the program that delivers the real-time schedule adherence information. The html that forms a fourth-level Web page includes anchors (the marks that tell a browser that some text is a link to another page) that have the user's route, direction of travel, bus stop and the Train number of their bus-of-interest explicitly stated. An example is:

```
<A HREF="/cgi-bin/TTIA.pl?17 Holgate&w&1171&Holgate and
                        Foster&10:05A">[4]
```

The route's name appears after the question mark. Separated from the route name by an ampersand is the day-of-the-week. A "w" stands for "weekdays." Tri-Met offers Saturday service as well as Sunday/Holiday service, but the restraints of rapid prototyping made a restriction to weekday service desirable. The route number and direction are encoded in the section after "w." Finally, the anchor includes the name of the bus stop and the time the bus is scheduled to arrive.

This format is simple-minded compared to the sophisticated Web pages of today. A commercial system would take advantage of html forms and include the Train number and the location identifier for the bus stop. More advice concerning the creation of a commercial system is offered in sections six and seven.

The end of the information collecting part of TTIA comes when the user clicks on a hyperlink like the one described above. The user-provided information is handed to a program called TTIA.pl. Through the mysterious mechanisms of UNIX[5] shell programming, this same information is handed to a Java[6] program which does the work of searching the database of exception messages and producing a Web page that gives the correct message to the user. The cgi-bin facility of the html server manages the transfer of information to TTIA.pl and the transfer of the Web page back to the user's browser. An attempt to port TTIA to a PC was enlightening. The cgi-bin facility for PC-based html servers is far more primitive than it is for Unix.

The Java program, named BDSApp.java, does some conversion of the user-provided time and checks the database of exception messages for any message concerning the bus-of-interest. Since TTIA is a prototype, the algorithm for storing and searching exception messages is unsophisticated. The messages are kept in a file and examined by a linear

[4] As the pages were refined, the html changed. This link is done with a form in the current version.
[5] The UNIX operating system predates DOS and other operating systems for home computers. Shell programming is a UNIX feature that can start and stop programs and manage input and output easily
[6] Java is a an object-oriented programming language designed to solve problems in the area of client/server programming.

search. The final task of the Java program is to refuse to deliver information that is suspect. In two situations information is withheld: if the user is asking about their bus too early, and if the user is asking about their bus too late. In both cases, there may be an exception message indicating the bus-of-interest is off schedule. But it is not useful to report that a bus due to arrive at the user's stop at 10:05 is a few minutes late at 8:00. Similarly, it is unimportant that the bus-of-interest is a few minutes late if the user is asking at 10:45. In other words, the exception message indicates only that a certain Train is off schedule. TTIA must decide if that information is pertinent to this user at this time.

The implementation of the TTIA system was influenced heavily by the requirements and advantages of rapid prototyping. Naturally, there were many challenges to overcome. The most interesting of these were those which we didn't anticipate. The next sections discuss the features of TTIA that were the result of the choice of rapid prototyping and the issues we faced that illuminated the task of real-time information delivery for a transit system.

6. TTIA: A Manageable Size

The production of a commercial system for delivering real-time schedule adherence information is no small task. Even the production of a prototype of such a system, such as TTIA, is a major undertaking. Reducing the task to a manageable size was one of the first and most significant challenges in the development of TTIA. Two strategies were employed. First, some features were implemented fully, but enabled for only some part of Tri-Met's system. Second, some attractive features of TTIA were delayed while the most fundamental features were implemented.

The implementation of TTIA for only seven of Tri-Met's one hundred routes and including only weekdays schedules reduced the number of static Web pages needed from 4500 to 160. Even 160 Web pages cannot be generated by hand effectively, especially when changes need to be made frequently. Therefore, the TTIA pages were generated by a perl[7] program. When changes were made to TTIA, the perl program was changed and the static pages regenerated. This process would have taken much more time for the full system. Also, the increased number of routes with some anomalous feature that required special attention would have been problematic.

To reduce the scope of the project, TTIA was first implemented without maps, without the module for helping riders find their bus stop (if it is not a time point) and without scrolling boxes or frames. This simpler version of TTIA was completely adequate for demonstrating the concept to potential users and other interested persons. The comments we received about the early TTIA guided our choice of features to include in the current TTIA prototype.

Some of the features that were not added to TTIA would be very useful for a commercial TTIA-like system. Under the heading of "rider training," for example, a component for helping users learn to read bus schedules would have been good to include. Our

[7] Perl is a language used for easily manipulating text, files and processes.

assumption that the users of TTIA are already bus riders and that they know which bus they want to catch freed us from writing any trip planning pages or programs. But a commercial system wouldn't ignore the new bus rider, the out-of-town visitor, the rider headed for some unfamiliar destination or the rider for whom schedules are indecipherable. Even sophisticated riders may need some clues about the TTIA information to avoid making mistakes. For example, if a rider uses TTIA and learns the his or her 10:05 bus is ten minutes late, he or she may decide to take the 10:25 bus instead. TTIA doesn't warn the user if the two busses don't serve all the same bus stops beyond the users place of boarding. The user is responsible for knowing exactly which busses are suitable. Thus an important recommendation is the inclusion of the overhead sign designation that provides users information about the bus destination.

Many other aids to the user were attractive but beyond the scope of the TTIA prototype. For example, TTIA requires the user to specify their route before specifying their direction of travel and bus stop. It was beyond the scope of the project to structure the pages in a way that would allow the user to indicate that he or she is interested in any bus that will travel inbound (for example) from a certain stop. A related modification would allow the user to state the information concerning route, direction and bus stop and ask when the next bus will arrive. For the user unfamiliar with the area they're in, TTIA could offer help locating a nearby bus stop. One feature included in the prototype allows the user to click on a time point of a schematic route map, displaying the schedule at that time point. Thus the user does not need to know the timepoint name. An improvement would be the facility to allow users to click anywhere along the bus route to indicate their stop. Better yet, smart maps would produce some result when the user clicked anywhere — even off the bus route. The map could zoom in from the schematic route map and display the selected area with a detailed street map showing individual bus stops as well as time points.

If a TTIA user wants to know each day whether a certain bus was on schedule, say the bus that they catch home from work, some kind of reminder could be communicated to them. Though this was originally within the scope of the TTIA project, it was not implemented because such a service would require some kind of subscription on the part of the user and, possibly, the distribution of software. A Web browser is a standard piece of software and the assumption that users have a browser doesn't limit TTIA.

Laying aside these particular ideas, the scope of TTIA was reasonable. Set aside were additional cosmetic improvements. Focus was on functionality and user-friendliness. User-friendliness dictates nice-looking pages, but beauty had to wait.

7. Issues and Obstacles

As TTIA was implemented, many unexpected problems arose and issues we had considered briefly came up again as decisions had to be made. Quite a few of these issues and obstacles will continue to be obstacles in any real-time information delivery project. In this section we introduce and discuss those problems and issues that are most important.

Our first attempt at a user interface adopted the structure of the Tri-Met schedule pages from the Web. Tri-Met displays a grid of times for which the rows are labeled with the timepoint names and the columns (though not labeled) correspond to distinct trips, each with its own train number. The users to whom we showed these pages found the information too dense. The choosing of a time point and a bus arrival time in one step required too much information to be displayed on a single page. Our solution was to break the choice of a time point and the choice of a bus arrival time into two steps. However, this greatly increased the number of static Web pages we needed.

Not only did we adopt the structure of the Tri-Met schedule pages, we used them as input to a perl script that generated the early TTIA pages. This had an unexpected effect. Since the Tri-Met pages are displaying the bus arrival times for use by a human, the individual times on the screen are not hot links — that is, they are not clickable. But the bus arrival times displayed by TTIA are hot links, and the information generated is used by a program. We learned that the kind of information we wanted had been omitted from the Tri-Met pages because riders don't need it. Specifically, we were missing the Train number for an individual bus arrival time and the location identifier for a bus stop. Had we begun with the Tri-Met database, we could have generated Web pages that included the Train number and location identifier very easily. In fact, the program used at Tri-Met to generate the schedule pages for the Web *does* use the output of a database query as its input. Using the Web pages as input to our Web-page-generating program was good for rapid prototyping, but not good in the long term.

Some of the peculiarities of the kind of data that represents a transit system cropped up as obstacles for TTIA implementation. A greater familiarity with the data that represents a bus system might have forestalled some trouble. For example, some bus stops are visited twice on a route. Working on the assumption that any bus stop is visited only once by any bus as it travels in one of the two directions on the route, the TTIA Web page generating code was written to place in the list of the times that a certain route serves a stop one entry for every Train number. More than one bus in our small sample looped, visiting a stop twice for a single Train number.

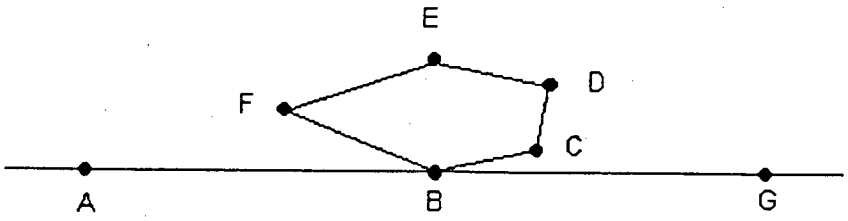


Figure 13: The bus stops at A, then B, C, D, E, F, then B and finally G

Our code should have assembled the list of times the bus passes stop B by collecting and sorting the *times* B is served by that route, not the Train numbers that serve B. The list of times must reflect that the next bus arriving at stop B may have the same Train number as the previous one (and is, in that case, the same bus).

In a similar vein, we didn't anticipate the significance of the fact that some bus routes follow different paths at different times of the day. For example, one bus may serve all the stops along a route from the city center to a distant suburb, but the next bus, which has the same route name and number, may end its service half-way along the route. Express busses produce the same effect. Some stops are part of the route, but the express bus doesn't stop at them. These anomalies have stranger companions. Some routes have alternate stops, so that one bus takes the path A, B, C, D, E, I and the next bus serving that route takes path A, B, F, G, H, E, I.

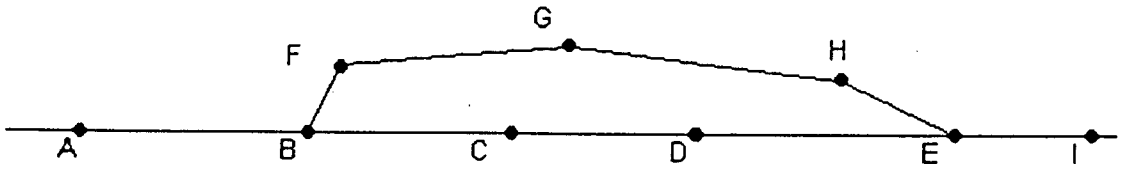


Figure 14: Two Paths on One Route

These situations pose a problem for TTIA when the user asks for help locating their bus stop. Which list of stops shall we give them to examine? If we give them all the stops, will users assume the order in which the stops appear in the list is the order in which they are served? We decided to allow users to examine the list of stops along the path served by the next bus (with respect to the time of day that they are making their request to TTIA). If they don't find their bus stop, they can request another list of stops to examine. More information about these decisions is given in Appendix A.

Enabling users to choose a bus stop between time points leads to a calculation of the time the bus will reach that bus stop based on the time the bus is due at the nearby time points. The Interpolated Time Module (ITM) calculates the arrival time of a bus at a non-timepoint stop. But routes that have several paths can complicate this. Consider a case in which I, D and H are time points, but E is not. If a user chooses E as their bus stop, which are the nearby time points, I and D? Or I and H? This is another example of the sometimes subtle challenges that arose as our implementation progressed.

To return to the decision to use static Web pages, there were several issues of interest. First, the 160 static pages have to be regenerated every time the bus schedules change. Major changes occur four times a year. This means that someone must be in charge of the

maintenance of any TTIA-like system, unless the pages are not static. If the pages are created dynamically from a database of route information, the changes in the data base will be enough to update the real-time system.

Second, the tree structure of the TTIA pages made it imperative that a user provide the information needed to deliver the message about their bus in the order we conceived. If a user wanted to indicate their stop first, the TTIA pages would not prompt them for the remaining information. One solution would be to use a more sophisticated algorithm for collecting information which would improve TTIA whether the pages were static or dynamic. If the pages were created dynamically from a database, however, it would avoid the explosion of static pages needed to allow the user to provide information in a different order than that which TTIA uses.

Third, we did find it desirable to add a dynamic component to the fourth level of static Web pages. For a bus that runs frequently during the day, the list of scheduled bus times that a fourth level page needs to display to the user is large — more than a hundred. To make the page more readable, we assumed that the user was interested in update information about a bus that is arriving at his or her stop soon after the time that the request is made. So the user's action, opening a URL, causes the execution of a perl program that filters the list of times on the static page. The page is displayed without the times that are more than 15 minutes in the past or 45 minutes in the future. Thus, the page is static but the page is edited dynamically.

On the topic of the real-time information itself, there were some significant obstacles. First, we had to decide how many exception messages to save for use by TTIA. Since another application at Tri-Met uses the exception message, there had been some planning for the transmission of exception messages on a local area network. Saving only one exception message for each Train number would allow us to report that a bus was running late, but would not allow us to deduce anything about the bus's situation. For example, an exception message that says that Train 1703 is 8 minutes late gives no clue about the bus's situation. But a bus reporting 8 minutes late which reports it is 10 minutes late and then 12 minutes late at 2 minute intervals is a bus that is not moving at all. We decided that deducing information based on the exception messages was beyond the scope of our project and thus, we saved just one exception message for each Train number.

The other issue concerning exception messages was the delays that have occurred in the installation of Tri-Met's Bus Dispatch System. Exception messages are collected primarily for the use of dispatchers who are monitoring and adjusting bus service moment to moment. The delivery of exception messages to TTIA had not been arranged as of this writing. The availability of several groups of test data made it possible to create a reasonable database of exception messages, but real-time data is not available.

Update pages were another cause of serious consideration. By showing TTIA to potential users, we learned that technical language had to be avoided in the user interface. For example, at first we used the phrase "estimated arrival time," but some users found "arrival" confusing since they were departing on their trip. The format we settled upon

doesn't use the estimated arrival time in a sentence. It is just displayed on the screen under the heading "Update Information."

Update pages were an issue with regard to disclaimers, too. When a bus is reported late, we need to alert the user that the bus may not continue to run late. The bus could be on time or even early to their stop. One problematic situation involves layovers at the end of a trip. If a bus is twelve minutes late, for example, as it nears the end of a trip but is scheduled for a 15 minute layover to give the driver a break, the bus may very well be on time as it starts the next trip. TTIA doesn't attempt to identify these situations, but delivers a disclaimer that may help users consider that possibility.

Another disclaimer is needed because an unsophisticated user might request information about their bus 15 minutes before the bus is scheduled to arrive and not ask again. Since a browser will continue to display the last Web page accessed, we were concerned that - user might think that an update report gave the latest information when, in fact, it was displaying the information delivered some time ago. To solve this problem, we added to the update page a statement of the time at which the page was assembled. Html mechanisms for solving this kind of problem are beginning to emerge. Client pull and server push are two mechanisms that allow the refreshing of Web pages without any action being taken by the user. These are possibilities for future real-time information delivery systems — especially those which are not interactive.

TTIA was designed not to deliver useless or misleading information. For example, if a user asks about the progress of his or her bus-of-interest which is scheduled to arrive at their stop at 5:03 p.m., the information delivery component of TTIA needs to check the current time. If it's currently noon, it would be wrong to report any information about the bus's schedule adherence. However, there could be an exception message in the data base for the Train number of the 5:03 bus (that bus may be serving some other route at noon — it has the same Train number all day). So we need to deliver an update page that tells the user that no information is known at this time.

Useless or misleading information is also a danger if the user asks about a bus that has already passed their stop. An exception message in the data base may show that the bus is 5 minutes early at the time of the user's request. If they are asking about a bus that was scheduled to arrive two hours ago at their stop, it doesn't mean that the bus was 5 minutes early then. So late requests must also be caught and answered with a reasonable message.

Reporting that a bus is early can also be useless or misleading information. A bus will more likely lose time if it is early than gain time if it is late. Possibly a real-time information system should treat early busses differently by giving a different disclaimer or by estimating the arrival time differently. A really sophisticated system might even account for the greater likelihood that a driver will run early right before a coffee break than at other times.

The TTIA system is intended to give information about small schedule deviations in normal conditions. When some unusual event occurs, TTIA may not give useful

information. For example, suppose snow falls in Portland, the city Tri-Met serves. Many busses go on snow routes. Some busses could be hours behind schedule. Riders may not want to know if their bus-of-interest is late as much as they want to know when the next bus will arrive at their stop. Or they may want to know how long it is taking busses on their route to arrive downtown (for example). TTIA is not able to deliver this information. A commercial system could be written to do so, though. We learned, as TTIA was developed, that there are many extensions that would make any deliverer of real-time information more useful. It was crucial to be clear about the information we had to report and the information we didn't have.

The Bus Dispatch System faces problems on a snow day in Portland, too. Since most busses will be running late, the Bus Dispatch System will be flooded with schedule exception messages. In order to reduce the number of messages, the dispatchers can change the parameters which control the definition of lateness. So, for example, the dispatchers may decide they need exception messages only for those busses running more than 30 minutes late. This effects the information available to TTIA. If a user requests update information for their bus-of-interest and receives an update report saying that the bus has not been reported late, they may construe that to mean that the bus is on time - when in fact it is 29 minutes behind schedule.

Another problem arises from the absence of an exception message. A bus will send an exception message only if it has on board a PCMCIA card (a removable computer component) which carries the information about the schedule. But the bus with the card aboard may break down and be replaced by another bus. Then there will be no exception messages for that bus, as it will have no information to use to determine its schedule adherence. Worse still, the original bus may continue to send exception messages saying it's 2 or 3 hours late! We don't want TTIA to make claims in this situation.

The same argument shows that TTIA ought not to claim that a user's bus-of-interest is on time if that bus isn't running at all. Suppose a bus never leaves the garage because there isn't a driver available. TTIA will look for an exception message, find none and report that the bus "is not known to be late." The user may well interpret this to mean the bus is on time, when there is no bus coming.

These troubles could be addressed if TTIA had access to more information about each bus. There are messages other than exception messages that are communicated to the Bus Dispatch System from a bus. For example, there are "health messages" that a bus sends regardless of its schedule adherence status. If a bus had never departed to serve the trip which the user has chosen as their bus-of-interest, there would be no health message from that bus. The creators of the Bus Dispatch System anticipated these kinds of needs, so there are also broken-down bus messages, reroute messages and other kinds of information that would benefit a complex real-time information delivery system.

Our experience developing the TTIA system suggests that either a simple system or a complex system can be useful. A simple system requires good disclaimers or restriction to sophisticated users. An important outcome of this project is the insight that many of the

issues and obstacles we encountered have to be addressed as thoroughly for a simple system as for a complex system.

8. Producing a Real-Time Information Delivery System

Producing a commercial TTIA-like system requires the planning and management that any software project requires. In this section we offer advice about project components that are specific to real-time applications and project components that are specific to transit applications.

The feasibility of an a real-time Web-based implementation has been demonstrated by the TTIA project.

The scope is one of the major concerns of an implementer. Good planning discourages time-consuming additions that are outside the original design of the system. A decision should be made early about the features that a TTIA-like system will include and exclude.

The choice of an implementer should be made with attention to the familiarity of the person with the data for the transit system. Some features of the data describing a bus system are anything but obvious to the uninitiated. Thus a transit expert is an asset worth finding.

If the organization contemplating a TTIA-like system has not offered any other real-time services to the public on the Web, network connections should be a concern early in the project. Time and expertise are needed to ensure a reliable and secure connection to the transit provider's data.

This document describes many facets of our experience with TTIA's development. We learned, for example, that static pages are not a good choice for any system that's not small. The insights provided may steer other projects and contribute to avoiding problems. As real-time systems are certainly going to be an increasing part of Web services in the future, the difficulties and expediencies of TTIA may provide guidance.

Appendix A: The Interpolated Time Module(ITM) of TTIA

Jun Qiu

The Interpolated Time Module (ITM) is TTIA's database module. One of the important tasks of this module is the interpolation of bus arrival times at the bus stop level, dynamically, from the times at the time points (a subset of bus stops) level. Because the user interface is a Web module, the ITM will solve the problem of the integration of the ITM module with the Web Module. Then, the ITM module calculates the bus arrival time at the stop level. This appendix documents solutions to these two problems: the interface and the calculation.

Data Transfer Specification for the Interpolated Time Module

The ITM module receives requests from TTIA and creates results to send back to TTIA. The TTIA prototype includes an HTML page offering time points for the users to choose among. If the user cannot identify a time point near his or her bus stop, he or she may either go to a map to facilitate making the choice or go to a help page. In this latter case, the user clicks on the highlighted word *help* and a cgi-bin program is executed. That program sends requests to the Interpolated Time Module and receives the estimated scheduled time at a specific bus stop.

The transfer of data is handled differently on a Unix machine than on a PC. This paper will establish a temporary process for data transfer which relies on files. That is, the data going to the Interpolated Time Module will be written to a file, and the data returned by the Interpolated Time Module will be written to a file. These files will have names. The programs dealing with these files will need to know the exact name of the file. This is poor programming practice. The newest version of MS ACCESS (ACCESS 97) will have a built-in mechanism for transferring data between the Internet and ACCESS.

Initial Bus Stop List Request

Input

Input to the Interpolated Time Module at the point called Initial Bus Stop List Request:
a string in the file **ITMin1.txt**
example: **route=008&day_of_week=w&direction_of_travel=1&count=1**

fields:

route specifies an individual bus route
day_of_w specifies one of the three kinds of days: weekdays (w), Saturdays (s) and Sundays and holidays(h).
direction_of_travel specifies the direction of the route. "1"- Inbound, "0"- Outbound

Output

Output from the Interpolated Time Module: text in the file **ITMout1.txt**
example:

count = 1
8 1 1 3288 "KIRBY" "HAYDEN MEADOWS"

```
8  1  1   8148  "KIRBY"                  "HAYDEN MEADOWS"
8  1  1   2646  "HAYDEN MEADOWS DR"      "G I JOES"
8  1  1   2649  "HAYDEN MEADOWS DR"      "WHITAKER"
8  1  1   6256  "WHITAKER"               "BURGER KING"
8  1  1   1298  "DENVER"                 "COLUMBIA"
8  1  1   1307  "DENVER"                 "WILLIS"
8  1  1   1297  "DENVER"                 "KILPATRICK"
8  1  1   1304  "DENVER"                 "WATTS"
8  1  1   1301  "DENVER"                 "TERRY"
8  1  1   1299  "DENVER"                 "RUSSETT"
8  1  1   3489  "LOMBARD"                "FENWICK"
8  1  1   3506  "LOMBARD"                "INTERSTATE"
```

The first line of the output file contains the count. The count is 1 when a user makes the first request for a bus stop list. In the following lines, the first column gives the route number. The second column gives the direction. The third column gives the path number. The fourth column gives location ID. The fifth column gives the name of street on which the bus stop is located. The last column gives the cross-street that, with the previous information, identifies the bus stop. The file is a delimited text file.

The information returned is displayed to the user. The user is offered a list consisting of these bus stops and asked to either choose among them or indicate that he/she wants to see another list for another path. One bus route may have several different paths. Busses follow different paths at different times of the day or run alternatively on two or more paths. For example, on many routes, short-lined busses alternate with those going the full route distance.

Additional Bus Stop List Request

If the user indicate that she/he did not find the desired bus stop on the list of stops displayed on the web page, then the following input and output occurs.

Input

Input to the Interpolated Time Module at the point called Additional Bus Stop List Request:
a string in the file **ITMin2.txt**

example: **route=008&day_of_week=w&direction_of_travel=1&count=2&path=1&path=2**

fields:

route specifies an individual bus route
day_of_w specifies one of the three kinds of days: weekdays (w), Saturdays (s) and Sundays and holidays(h).
direction_of_travel specifies the direction of the route. "1"- Inbound, "0"- Outbound.
count specifies the number of requests for bus stop lists the user has made before this request. Each request creates output of a list of stops for a path.
path specifies a path which the user examined before this request and in which the user did not find his/her bus stop. There can be more than one **path** part of the input string. The number of **path** parts is indicated by the previous **count** value. All Path values should be unique.

Output

Output from the Interpolated Time Module:
one of the two types of text in the file **ITMout2.txt**

Two possibilities exist for output. The first is a single-line file containing the three words "no more paths." The second is much like the previous step's output.

example:

count = 2

8	1	2	3288	"KIRBY"	"HAYDEN MEADOWS"
8	1	2	8148	"KIRBY"	"HAYDEN MEADOWS"
8	1	2	2646	"HAYDEN MEADOWS DR"	"G I JOES"
8	1	2	2649	"HAYDEN MEADOWS DR"	"WHITAKER"
8	1	2	6256	"WHITAKER"	"BURGER KING"
8	1	2	1298	"DENVER"	"COLUMBIA"
8	1	2	1307	"DENVER"	"WILLIS"
8	1	2	1297	"DENVER"	"KILPATRICK"
8	1	2	1304	"DENVER"	"WATTS"
8	1	2	1301	"DENVER"	"TERRY"
8	1	2	1299	"DENVER"	"RUSSETT"
8	1	2	3489	"LOMBARD"	"FENWICK"
8	1	2	3506	"LOMBARD"	"INTERSTATE"
8	1	2	3451	"LOMBARD"	"#932"
8	1	2	3453	"LOMBARD"	"ALBINA"
8	1	2	3513	"LOMBARD"	"KERBY"
8	1	2	5998	"VANCOUVER"	"LOMBARD"
8	1	2	5990	"VANCOUVER"	"HOLLAND"
8	1	2	5983	"VANCOUVER"	"BUFFALO"
8	1	2	5981	"VANCOUVER"	"BRYANT"
8	1	2	5986	"VANCOUVER"	"DEKUM"
8	1	2	6002	"VANCOUVER"	"PORTLAND"
8	1	2	4488	"PORTLAND"	"WILLIAMS"
8	1	2	4476	"PORTLAND"	"MALLORY"
8	1	2	5903	"M L KING"	"DEKUM"
8	1	2	1293	"DEKUM"	"6TH"
8	1	2	3644	"MADRONA"	"DEKUM"
8	1	2	1265	"DEKUM"	"DURHAM"
8	1	2	1263	"DEKUM"	"BELLEVUE"
8	1	2	1264	"DEKUM"	"DEAN"
8	1	2	6794	"15TH"	"JUNIOR"
8	1	2	6804	"15TH"	"PORTLAND"
8	1	2	6772	"15TH"	"AINSWORTH"
8	1	2	6793	"15TH"	"JARRETT"
8	1	2	6796	"15TH"	"KILLINGSWORTH"
8	1	2	6782	"15TH"	"EMERSON"
8	1	2	6813	"15TH"	"SUMNER"
8	1	2	6774	"15TH"	"ALBERTA"
8	1	2	6820	"15TH"	"WYGANT"
8	1	2	6788	"15TH"	"GOING"
8	1	2	6805	"15TH"	"PRESCOTT"
8	1	2	6802	"15TH"	"MASON"
8	1	2	6807	"15TH"	"SHAVER"
8	1	2	6784	"15TH"	"FAILING"
8	1	2	6776	"15TH"	"BEECH"
8	1	2	6786	"15TH"	"FREMONT"
8	1	2	6798	"15TH"	"KLICKATAT"
8	1	2	6809	"15TH"	"SISKIYOU"
8	1	2	6812	"15TH"	"STANTON"
8	1	2	6800	"15TH"	"KNOTT"

8	1	2	6778	"15TH"	"BRAZEE"
8	1	2	6816	"15TH"	"THOMPSON"
8	1	2	6817	"15TH"	"TILLAMOOK"
8	1	2	6779	"15TH"	"BROADWAY"
8	1	2	6789	"15TH"	"HALSEY"
8	1	2	6818	"16TH"	"MULTNOMAH"
8	1	2	4045	"MULTNOMAH"	"13TH"
8	1	2	4044	"MULTNOMAH"	"11TH"
8	1	2	4056	"MULTNOMAH"	"9TH"
8	1	2	4054	"MULTNOMAH"	"7TH"
8	1	2	4043	"MULTNOMAH"	"GRAND"
8	1	2	9242	"MULTNOMAH"	"M L KING"
8	1	2	1098	"ROSE QUARTER TC"	"B2"
8	1	2	9311	"GLISAN"	"3RD"
8	1	2	9304	"5TH"	"FLANDERS"
8	1	2	9301	"5TH"	"DAVIS"
8	1	2	7589	"5TH"	"BURNSIDE"
8	1	2	7626	"5TH"	"OAK"

fields:

The first line of the output file contains the count. The count is (1) when the user has made one request for a bus stop list. In the following lines, the first column gives the route number. The second column gives the direction. The third column gives the path_number. The fourth column gives the LOC_ID. The fifth column gives the street on which the bus stop is located. The last column gives the cross-street that, with the previous information, identifies the bus stop. The file is a delimited text file.

Again, the user can either identify her/his stop from this list or ask for another list of stops. This is a looping process. The loop continues until the user finds the stop she/he wants.

Time at Stop Request

After the user selects a bus stop (for example, **"15TH" & "KILLINGSWORTH"**), its location ID (6796) is used as input to the Interpolated Time Module.

Input

Input to the Interpolated Time Module at the point called Time at Stop Request:

a string in the file ITMin.txt.

example: **route=008&day_of_week=w&direction_of_travel=1&locid=6796**

fields:

route specifies an individual bus route

day_of_w specifies one of the three kinds of days: weekdays (w), Saturdays (s) and Sundays and holidays(h).

direction_of_travel specifies the direction of the route. "1"- Inbound, "0"- Outbound

locid specifies the location of a bus stop that a user has chose.

Output

Output from the Interpolated Time Module: text in the file **ITMout3.txt**

example:

14:20	1500	2	805	85	3644	"0:03"	"NE 15TH AVENUE TO PORTLAND"
14:35	1510	7	809	85	3644	"0:03"	"NE 15TH AVENUE TO PORTLAND"
14:50	1540	2	807	85	3644	"0:03"	"NE 15TH AVENUE TO PORTLAND"
15:04	1550	7	801	85	3644	"0:03"	"NE 15TH AVENUE TO PORTLAND"

15:15	**1560**	**2**	**6068**	**85**	**3644**	**"0:03"**	**"NE 15TH AVENUE TO PORTLAND"**
15:26	**1570**	**7**	**811**	**85**	**3644**	**"0:03"**	**"NE 15TH AVENUE TO PORTLAND"**
15:37	**1580**	**2**	**810**	**85**	**3644**	**"0:03"**	**"NE 15TH AVENUE TO PORTLAND"**

Fields:

The first column gives the stop time, which is the result of this module.
The second column gives the trip number.
The third column gives the path number
The fourth column gives the train number
The fifth column gives the up stream time point number which is unique to each time point.
The sixth column gives the up-stream time point **LOC_ID**, which is unique to each location (stop or time point).
The seventh column specifies the minutes that the bus travel from the upstream time point to the user's stop.
The last column gives the overhead sign text of that trip.

Although the above result shows that the up-stream time point for this specific user entered stop is unique, this is not always true. Thus, it is appropriate to generate the column of information about the up-stream time point. We notice that in this time range, busses run on path 2 and path 7 alternately. However, on both paths, busses pass by that stop (**"15TH" and "KILLINGSWORTH"**).

Database Design for the Interpolated Time Module

The Interpolated Time Module (ITM) of TTIA is a relational database created on MS ACCESS. Source data tables are from Tri-Met. Two issues are discussed in this section: database tables and database relations.

Creating Database Tables

TTIA is supported by a large amount of data in the Interpolated Time Module. The data is separated into several tables. (* key items for creating relations)

1. Bus stop table

Item Name	TYPE
* SERVICE_TYPE	T
* RTE	N
* DIR	N
* PATH_NUMBER	N
* LOC_ID	N
STOP_DISTANCE	N

SERVICE_TYPE:	Describes which day of week a bus runs. (In this table it is blank. It will be populated in combination with trip information.)
RTE	Bus Route.
DIR	Direction of travel for a bus.
PATH_NUMBER	A number identifying the path, (i.e., pattern). One route has multiple paths.
LOC_ID	A unique number for each location where there is a bus stop (including time point).
STOP_DISTANCE	Distance to a stop from the beginning of a route.

Each bus stop is keyed by multiple items: "Rte, Dir, PATH_NUMBER, and LOC_ID". The value of LOC_ID is unique for each of the stop. Several routes may pass one stop. SERVICE_TYPE means the day of week a bus runs, including weekdays, Saturdays, or Sundays and holidays. This table creates such a blank column. This column will be populated with the information from trip table (below).

2. Time point table

Item Name	Type
* RTE	N
* DIR	N
* PATH_NUMBER	N
* TIME_POINT	N
LOC_ID	N
STOP_DISTANCE	N

RTE	Bus Route.
DIR	Direction of travel for a bus.
PATH_NUMBER	A number identifying the path, (i.e., pattern). One route has multiple paths.
TIME_POINT	A unique number to each time point.
LOC_ID	A unique number for each location where there is a bus stop (including time point).
TP_DISTANCE	Distance to a stop from the beginning of a route.

Time point is keyed by "Rte, Dir, Path_number, and Time_point".

3. Trip table

Item Name	TYPE
* SERVICE_TYPE	T
* RTE	N
* DIR	N
* PATH_NUMBER	N
TIME_POINT	N
* TRIP_NUMBER	N
TIME	N

SERVICE_TYPE:	Describes which day of the week a bus runs.
RTE	Bus Route.
DIR	Direction of travel for a bus.
PATH_NUMBER	A number identifying the path, (i.e., pattern). One route has multiple paths.
TIME_POINT	A unique number for each time point.
TRIP_NUMBER	A number describing a specific trip on a route and direction.
TIME	Scheduled time for a bus to arrive at a time point.

Trip is keyed by the composite items "RTE, DIR, PATH_NUMBER, TRIP_NUMBER". One trip passes several time points. The trip table includes the important columns of scheduled TIME and SERVICE_TYPE. The scheduled TIME is at time point level. The service type includes weekday, Saturday, Sunday/holiday. The information in this table will be used for filling in the blank column in the Bus Stop Table.

4. Stop name table

Item Name	TYPE
* LOC_ID	N
LOCATION	T
INTERSECTION	T
X COORDINATE	N
Y COORDINATE	N

LOC_ID	A unique number for each location where there is a bus stop (including time point).
LOCATION	The street a stop is located on.
INTERSECTION	The nearest cross-street for that stop.
X_COORDINATE	X coordinate .

Y_COORDINATE Y coordinate.

Tri-Met identifies a stop name by the two nearest intersection's name. Each stop with one location-ID has a pair of street names to describe the stop: the street name and the intersection street's name. With location_ID field (LOC_ID) as the primary key, stop names is related to the bus stop table.

5. Train number table

Item Name	TYPE
* SERVICE_TYPE	T
* RTE	N
* DIR	N
* TRIP_NUMBER	N
* PATH_NUMBER	N
TRAIN_NUMBER	N

SERVICE_TYPE: Describes which day of the week a bus runs.
RTE Bus Route
DIR Direction of travel for a bus.
PATH_NUMBER A number identifying the path, (i.e., pattern). One route has multiple paths.
TRIP_NUMBER A number describing a specific trip: one route, one direction, and one day.
TRAIN_NUMBER:A number which identifies the vehicle (train) which is assigned to a set of trips.

"Train" is the Tri-Met term for a single vehicle to serve a set of trips in a single day. A TRAIN_NUMBER is assigned to a bus that serves a set of trips. A TRAIN_NUMBER is important when relating bus schedule times with other module. The ITM module will give TRAIN_NUMBER as one of the outputs. A train is keyed by a composite items "SERVICE_TYPE, RTE, DIR, PATH_NUMBER and TRIP_NUMBER".

6. Overhead sign table

Item Name	TYPE
* SERVICE_TYPE	T
* RTE	N
* DIR	N
* PATH_NUMBER	N
HEADSIGN	T

SERVICE_TYPE Describes which day of the week a bus runs.
RTE Bus Route
DIR Direction of travel for a bus.
PATH_NUMBER A number identifying the path, (i.e., pattern). One route has multiple paths.
HEADSIGN A text string which tells the bus's trip destination (and sometimes origin).

In this table, the overhead sign information is stored. The overhead sign is a text string displayed on each bus when the bus is on a trip. The sign informs the riders where the bus is headed. A unique overhead sign is also keyed by the composite items "SERVICE_TYPE, RTE, DIR and PATH_NUMBER".

Building Relations

The key items of each table are used for creating relations among tables. For calculating stop time, three tables are critical: the Bus Stop table, the Time Point table, and the Trip

table. The Bus Stop table tells which stop a user wants. The Trip table has the bus arrival time at time point level. The Time Point table bridges the former two tables. For other information, such as overhead sign, stop name, and train number, the other three tables are added to the relations (See Diagram 1). Relations are usually based on composite keys rather than a single key.

Interpolating Bus Arrival Time Dynamically

The database relations are used in this section. An earlier version to interpolate bus arrival time at the stop level showed that static interpolation would populate a large database. In this prototype, the ITM module employs a dynamic interpolation method.

Dynamic interpolation means that the interpolation happens based on user's input. In TTIA, user-entered data includes the day of the week, the route, the direction, the stop location, and the time a user wants take the bus. The ITM will use this information as query parameters. Based on the tables and the established relations, the interpolation will happen in a small range set by time, possible path, and the possible interval in a bus route (Diagram 2). Diagram 2 is a schematic that illustrates the interpolation process.

From the Bus Stop table, the program selects a subset of a complete path. To do this, the ITM executes a query for one RTE, one DIRECTION, and one PATH_NUMBER. The resulting subset is a single path which has a list of stops. There are 10 - 100 stops for each path, depending on the length of the path. The pseudo code is:

Select bus stop table
where route = user-entered-route
direction = user-entered-direction
path_number = the first possible path number (loop to exhaust other paths)

From the Time Point table, the same query as the above is executed. The resulting subset is the same path which has a list of time points. There are several time points for each path, depending on the length of the path.

Select bus time point table
where route = user-entered-route
direction = user-entered-direction
path_number = the same path_number as the above query uses

From the Trip table, in addition to the three parameters: a RTE, a DIRECTION, a PATH_NUMBER, the query parameters include SERVICE_TYPE and TRIP_NUMBER. The resulting subset is a trip on that day of week and on the same path. The column TIME in this subset illustrates the scheduled time of each trip at the time point level. The pseudo code is:

Select bus trip table
where servicetype = user entered (or defaulted as system time)
route = user entered route
direction = user entered direction
path_number = the same path number as the above two queries.
and trips are within a range of user-entered time (or defaulted as system time)

The subset of stops and subset of time points, which have the same RTE, DIR and PATH_NUMBER, contains a list of stops and a list of time points. Their relations can be created using the LOC_ID as the common item. Because some of the stops are also timepoints, there exists a one-to-one and one-to-zero relation between the two subset.

The subset of time points and the subset of trip, which have same RTE, DIR, PATH_NUMBER, contains a list of time points and a a list of time at time point level. The relation between the subset of time points and the subsets of trip is created on the TIME_POINT as a common column. Because some of the time points are not used for time scheduling for a trip, there exists, again, a one-to-one and one-to-zero relation (See Table 1).

Table 1 illustrates three entities: a path with a list of stops, a path with a list of time points and a trip with a list of times passing that path. The highlighted part of Table 1 illustrates an example of interpolation. The bus stop (6796, see LOC_ID column) is located between two time points (up-stream 85; down-stream 81; see TIME_POINT column). For trip 1540, we get an interpolated time to this stop: 53400 (14:50 PM).

For computing all possible bus arrival times at a stop, the ITM program loops to exhaust all possible paths and all possible trips in a given time range (e.g. half hour before and after the time a user wants). Within a loop, when the three subsets are found and the relation is established as that in Table 1, the ITM traverses the three joint subsets to find the appropriate interval in which the stop is located. The up-stream and down-stream time points are flagged. Then the calculation described in Diagram 2 happens. These steps are run by embedding SQL clauses into Access Basic, which is a programming language built in MS ACCESS.

The result usually indicates that several busses will arrive at one stop in a time range. The result also shows that busses may run on different paths of a route even they arrives at the same stop.

Conclusion

The first section of the above document discusses the data transfer issue. The transfer process is based on reading and writing text files from/to a Web module and the ITM using the disk as media. This is troublesome concerning the security and the case when there are multiple users and frequent query transactions. However, the document clarifies the input and output which the ITM accepts and produces.

In addition to the interpolated bus arrival time, output from ITM to the Web module include important information such as overhead sign text and the train number. The overhead sign information will be used to create a friendly user interface. The train number will be useful for comparing the scheduled time with the real-time. This is one of the TTIA's future tasks.

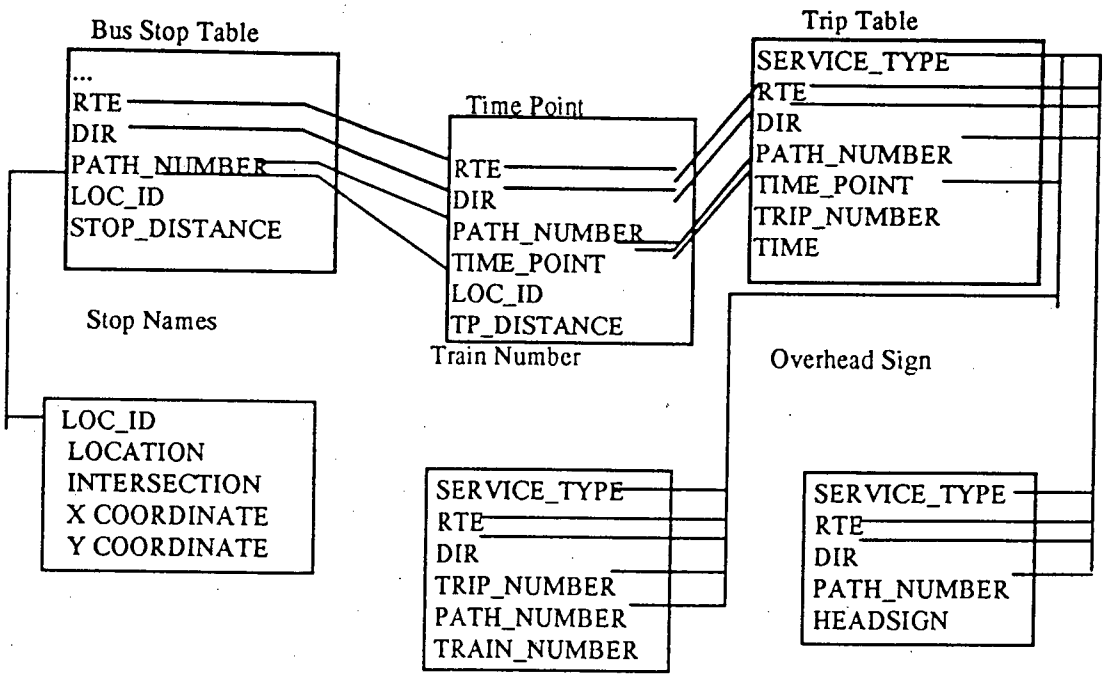


Diagram 1, Relations of Tables

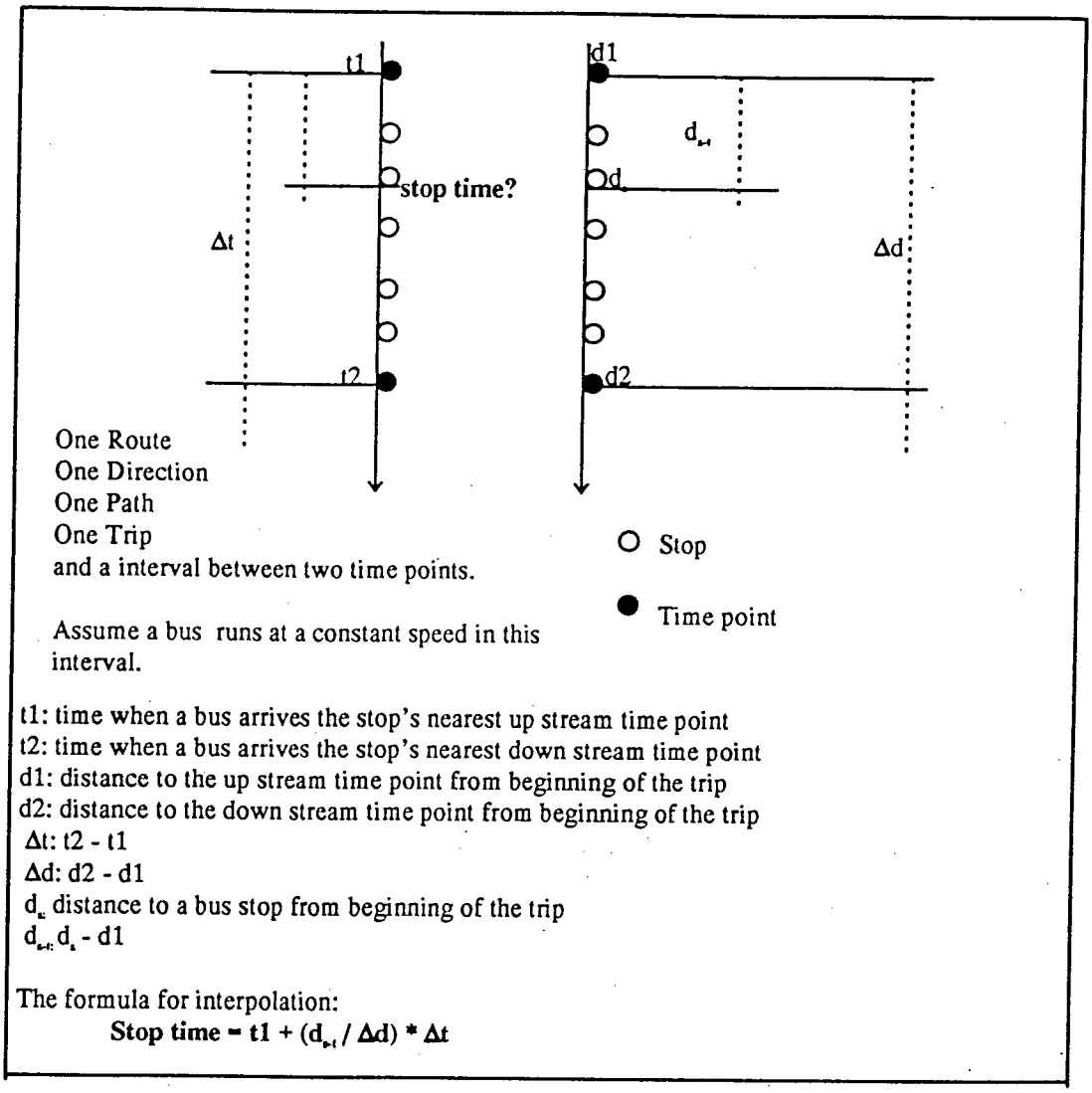


Diagram 2: Interpolation Model

This table shows the relations of the various tables.
Given a bus stop, the program searches for an interval. For example, given 9700 for the timepoint whose location id is 4543, the program finds the boxed interval.

RTE	DIR	PATH NUMBER	LOC ID	STOP DISTANCE (feet)	TIME POINT	TP DISTANCE (feet)	TRIP NUMBER	SERVICE TYPE	TIME (seconds from midnight)	TRAIN
109	1	1	8198	0.0000	9010	0	1510	W	54300	936
109	1	1	3123	787.1820						
109	1	1	7605	1328.4990						
Some Rows Deleted										
109	1	1	8458	10073.1795						
109	1	1	8743	10561.4555						
109	1	1	4508	11695.3495						
109	1	1	4621	13588.7735						
109	1	1	4523	15065.3725						
109	1	1	4617	15980.8997	8201	15750.6887	1510	W	55020	936
109	1	1	5818	16371.2957						
Some Rows Deleted										
109	1	1	4586	28640.9022						
109	1	1	4583	29075.1902						
109	1	1	4581	29653.7892						
109	1	1	4579	30300.4342						
109	1	1	4578	30606.1292						
109	1	1	4576	31115.4162						
109	1	1	4575	31828.4472						
109	1	1	4572	32273.1982	1094	32273.2181	1510	W	55680	936
109	1	1	4570	33105.2801						
109	1	1	4569	33761.5501						
109	1	1	4566	34935.6921						
109	1	1	4503	35958.7401						
109	1	1	4565	37000.1881						
109	1	1	4563	37582.2311						
109	1	1	4561	38046.5001						
109	1	1	4543	38896.0301	9700	38837.3011	1510	W	55980	936
109	1	1	4709	40320.4531						
109	1	1	4706	40867.0644						
109	1	1	4700	41577.1654						
109	1	1	4698	42316.6444						
109	1	1	4695	42985.6381	1093	42945.5101	1510	W	56220	936
109	1	1	4692	43762.3301						
109	1	1	4690	44505.8051						
109	1	1	4688	45092.7201						
109	1	1	4684	45841.9781						
109	1	1	4683	46428.8621						
109	1	1	4681	46683.1951						
109	1	1	4678	47772.9521						
Some Rows Deleted										

Appendix B: Feasibility of Placing Bus Route Maps on the Internet

Ray Jackson

This section of the report will outline the methods and issues that need to be addressed for the successful delivery of maps via the Internet. The maps are to show portions of Tri-Met's bus routes and street maps in the vicinity of bus stops. The report also presents a discussion of a first iteration implementation of the concept. This is but one aspect of a project being conducted by the Center for Urban Studies, Portland State University, titled *Evaluation of Actual Bus Arrival Time Information in Transit Service Reliability*.

This portion of the project is a feasibility study to determine what resources are needed to meet the goal of map creation and delivery via the Internet to a web browser. Other parts of the project are implementing other facets of providing bus arrival time information via the Internet to transit riders.

The current product of the study is a web page that leads users through the selection of bus route, direction and time of day, to produce a page that informs them whether or not the specified bus is running late. The data for the schedule adherence is based on exception reports generated by the busses themselves, and are radioed back to Tri-Met's bus dispatch center. The main report details the building of the web pages.

A logical extension to these pages is the addition of maps depicting bus routes and stops. First iterations will demonstrate the feasibility by using 'dumb' or raster graphics of the bus routes. Successive iterations will build upon this by integrating more detail and making the maps 'smarter'. By smarter, we mean that a user of the web page will be able to click on a location on the map, and the system will report scheduled deviation for that location. Smarter might also mean zooming to a location along a route for more detailed map showing the local street system and individual bus stops.

Before we describe the specifics of possible implementations, it will be useful to review the solutions others have presented. Because the concept of placing maps on the Internet is a recent one, few detailed reports are available. As expected most, if not all, geographical information systems vendors have rushed to produce a product that addresses this market, and have produced the requisite white paper to extol the virtues of their solution. Most have sample maps available on their web pages. A few examples, current as of spring 1997, are:

http://www.universal.ca/product/internet.html
http://www.caliper.com/MapServ
http://www.mapinfo.com/events/prosrv/webbank2.html
http://maps.esri.com/ESRI/arcview/demos.htm
http://maps.esri.com/ESRI/mapobjects/demos.htm

One of the concerns with this type of product is the rapidity of change in the marketplace. It is difficult to get detailed information regarding future products and their availability.

This can impact the planning aspects to a great deal. For instance, Intergraph recently announced their product, GeoMedia Web, between iterations of this report.

The topic has captured the attention of not only the vendors, but of academics and local government officials. It is of interest to the governments as a method of serving their populace, while at the same time reducing their costs to provide the information.

Several papers were presented at a regional GIS users group meeting in the fall of 1996 that also addressed this issue. Peterson (1996) discusses the approach taken by the EPA with their SITEINFO application. In this application, the user fills out a form requesting information in the vicinity of a city, and then the system performs in basically a batch mode to produce the necessary maps. Once finished, the user is sent an email message informing them that their maps and report are ready. This method involves little interaction between the user and the program beyond filling out the form. There is no dynamic interaction occurring, rather the user needs knows a priori what it is they wish to have mapped.

Ciscell (1996) describes his procedures for using Arc/Info to place maps on the Internet. He discusses using image maps to make a static map more accessible to the web page visitor, by allowing the user to select regions of the map that they are interested in. This is the next step in adding interaction to Internet map sites. This is akin to the various sites on the Internet that allow the perusal of street maps for the country.

A third paper (Pearson & Quetel, 1996) introduces ESRI's new solution, Map Objects Internet Map Server. Using a beta version of the software, they describe how the product would place a map on the Internet. This approach is the most dynamic of the three. In it the user is allowed to specify not only the area of a map to examine, but also what techniques should be performed on that area.

These three papers represent the continuum of how maps, and the information they provide, are being integrated on the Internet. At one end is the batch mode, where there is little user interaction with the maps. At the other end, near real-time creation of maps and delivery allow users to interact as if the maps were being generated on their own machines. This continuum extends from the methods used to way the information is presented to the user. From static forms and maps, to dynamic maps, there is a range of possibilities. The table below details some of the main features, and how the different techniques fare when compared to each other.

	Form Based	Static Maps	Dynamic Maps
Ease of Implementation	Easy	Simple	Harder
User Friendliness	Varies	High	High
Data Exploring	Low	Medium	High

There has been a shift in the last several months away from any sort of static presentation of data, to a more dynamic, user involved presentation. This can be witnessed by the explosion in the number of web pages that provide maps that users can select what it is they are interested in. These maps rely on recent advances in Internet distribution

This can impact the planning aspects to a great deal. For instance, Intergraph recently announced their product, GeoMedia Web, between iterations of this report.

The topic has captured the attention of not only the vendors, but of academics and local government officials. It is of interest to the governments as a method of serving their populace, while at the same time reducing their costs to provide the information.

Several papers were presented at a regional GIS users group meeting in the fall of 1996 that also addressed this issue. Peterson (1996) discusses the approach taken by the EPA with their SITEINFO application. In this application, the user fills out a form requesting information in the vicinity of a city, and then the system performs in basically a batch mode to produce the necessary maps. Once finished, the user is sent an email message informing them that their maps and report are ready. This method involves little interaction between the user and the program beyond filling out the form. There is no dynamic interaction occurring, rather the user needs knows a priori what it is they wish to have mapped.

Ciscell (1996) describes his procedures for using Arc/Info to place maps on the Internet. He discusses using image maps to make a static map more accessible to the web page visitor, by allowing the user to select regions of the map that they are interested in. This is the next step in adding interaction to Internet map sites. This is akin to the various sites on the Internet that allow the perusal of street maps for the country.

A third paper (Pearson & Quetel, 1996) introduces ESRI's new solution, Map Objects Internet Map Server. Using a beta version of the software, they describe how the product would place a map on the Internet. This approach is the most dynamic of the three. In it the user is allowed to specify not only the area of a map to examine, but also what techniques should be performed on that area.

These three papers represent the continuum of how maps, and the information they provide, are being integrated on the Internet. At one end is the batch mode, where there is little user interaction with the maps. At the other end, near real-time creation of maps and delivery allow users to interact as if the maps were being generated on their own machines. This continuum extends from the methods used to way the information is presented to the user. From static forms and maps, to dynamic maps, there is a range of possibilities. The table below details some of the main features, and how the different techniques fare when compared to each other.

	Form Based	Static Maps	Dynamic Maps
Ease of Implementation	Easy	Simple	Harder
User Friendliness	Varies	High	High
Data Exploring	Low	Medium	High

There has been a shift in the last several months away from any sort of static presentation of data, to a more dynamic, user involved presentation. This can be witnessed by the explosion in the number of web pages that provide maps that users can select what it is they are interested in. These maps rely on recent advances in Internet distribution

methodologies, away from the limited HTML-based browser views, to Java applets that provide for a wider range of customization. For the purposes of this report, we will lean toward the more real-time aspects.

As this study is using the Tri-County Metropolitan Transit District's (Tri-Met) bus system as the model, it is appropriate to examine what this agency has produced for their own web pages (http://www.tri-met.org). Tri-Met currently provides on their web page schedules for all their bus routes. They are adding to each bus route, a version of the route map as it appears from the printed schedule. This provides a schematic of what streets the bus routes follow, and shows time points and connections to other routes. The map is 'dumb', that is there is no link between it and the data that is provided.

Discussions with Tri-Met staff about the web pages have indicated that this is the level of detail they are currently willing to achieve. To create maps of greater detail and make them 'active' would require more resources and time than the perceived benefits would suggest is prudent. However, a useful extension would be to add buttons at the timepoints, linking the static maps with the underlying time information.

Product Selection

The proper product selection is based on a number of factors: Reliability, ease of use, speed, support, and ease of moving existing data are the main ones. For this project, this selection process has been circumvented due to existing investments in one vendor's previous products.

Issues

The issues surrounding the design and implementation of the web pages may be broken down by the stage of implementation: current prototype and future iterations. Most of the short-term issues have been or will be addressed by decisions made during the implementation. These include how the maps are to be integrated with the web pages and where the schedule information will be placed in relation to them.

Future implementation issues revolve around issues of efficiency. There is a possible jump in map size when the project moves from the 'dumb' graphics to the 'active' maps, which provide more detail. The number of maps needed for the first version is two per complete bus route. Individual maps would be needed to show each direction of each segment due to the way the maps would be linked with the underlying web pages with schedule information. While the individual maps of the bus routes are currently small, on the order of 10 kilobytes, a full fledged system of fifty bus routes, with two maps for each direction, and two maps for each segment would consume approximately two megabytes of space. This is for the 'dumb' graphics version. The 'active' version is likely to require more.

Placing a useful map on a web page is not always a straightforward operation. While a 'dumb' graphic representation of a map may be useful for some purposes, in other instances an 'active' map is the correct solution.

How long does it take to create a map on the fly? If the time is too long, will users decide to forego the maps, and conversely, is quick response a good thing? If map creation and download to a client web browser is very fast, will that user over consume the resources by examining more than just the page they are interested in?

Implementation

For the first iteration, an image map editor was used to construct the image maps needed to allow the clicking on the time points to obtain the schedule information. Currently, image maps have been produced for the seven routes used in this phase of the study. The files depicting the maps are from Tri-Met's web page and were downloaded for processing. A shareware program, MapEdit, was used to define the locations of the buttons used to direct the web browser to the page with the associated time information. For this implementation, buttons were added at only the time point and transfer center locations depicted on the map. These point to specific web pages that contain the schedule information for the bus route at that point (figure 1).

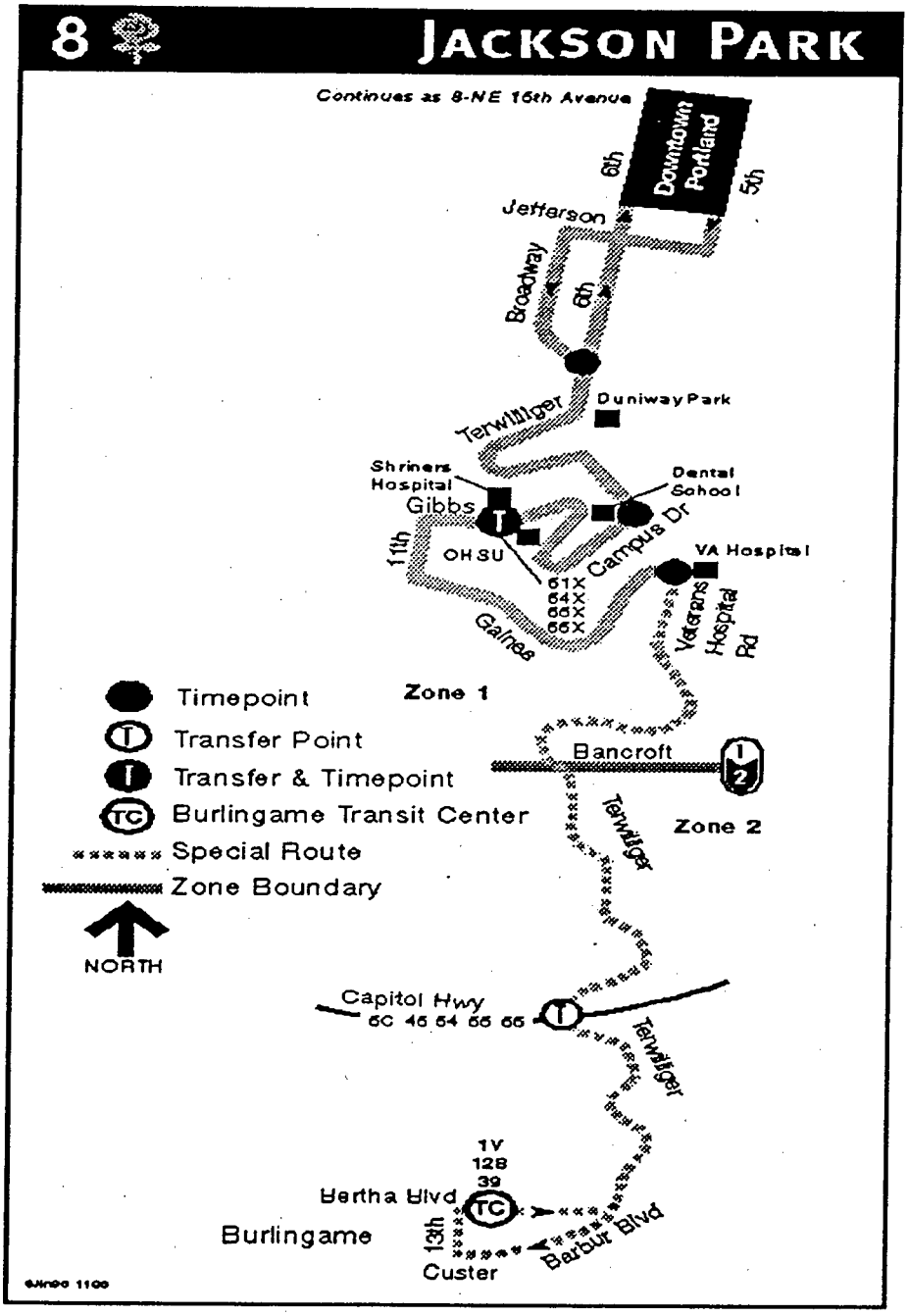


Figure 1: Map for Web Page Showing Timepoints as "Hot" Buttons

The request for the map is generated from a "button" on the web page showing some other route specific information. The first map generated should be a total route map that is showing the beginning and end points for the specific bus route depicted on the web page. This map will have the bus route, stops, and major arterials. Time points should be a different color to be easily distinguishable.

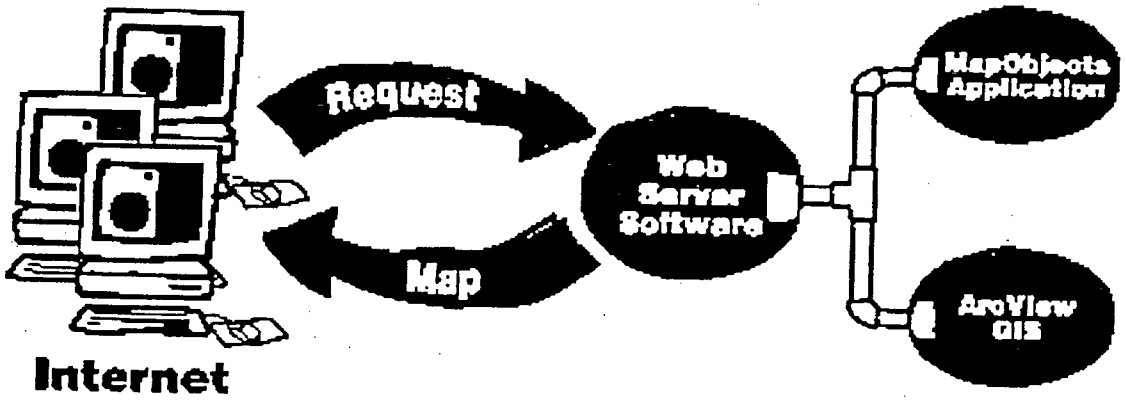
The user will then click on either a time point, or on a location between the time points. While in the first iteration only time points will produce a listing of bus times, in the subsequent iterations the areas between time points will signal to the controlling program that a new map needs to be generated. This map will be centered on the area clicked and will present a zoomed in view of the surroundings, with more detail of the cross streets so the user may identify where the bus stop is actually located.

The more feature-laden implementations will be achieved by creating an application that will take requests from the web client and produce a map that is centered on the area specified. This application may either be written with Microsoft Visual Basic (VB) using Map Objects from Environmental Systems Research Institute (ESRI), or the pre-existing functionality of ArcView may be used.

Visual Basic is chosen because it will allow for rapid prototyping of the application. It is also well supported in the Map Objects package in terms of samples and example code. Map Objects is chosen as it allows use of preexisting ArcView coverages, shapefiles, of the bus routes and road networks.

It is now possible to implement this in ArcView Internet Map Server (IMS), which will be shipping 'soon' from ESRI. There are advantages and disadvantages to each of these strategies. While Map Objects will allow for a more efficient GIS server, capable of handling more response with better throughput, it will also require writing the necessary applications from scratch, or modifying preexisting ones. For this implementation, this would not be a complex task. The advantage of ArcView Internet Map Server is that, supposedly, the application would not have to be written, but that the IMS would 'serve' an interface of the program to the web browser, thus allowing the user to utilize the familiar tools found in ArcView. The disadvantage of this approach involves the efficiency issue. ESRI representatives state that for sites with many 'hits' that ArcView IMS would not be able to handle as many requests as Map Objects with IMS.

For performance, the first maps will be pre-generated graphic files. These will provide a schematic representation of the bus routes and the time points. When the user selects an area that is between the time points, then a map depicting the bus route and stops for that area will be generated by the GIS backend and sent to the web browser through the connection method. This depends on the software being used. If the GIS server is running either ArcView or Map Objects, then the most likely implementation strategy will be to use ESRI's Internet Map Server. This product performs all the functions needed to link the requests from a web browser and server, with the GIS server. (see diagram below, taken from ESRIs web site)



The second method would be to use CGI (common gateway interface) to take the requests for the web server and translate them to requests that the GIS server can understand. This interface would also take the results from the GIS server and place them where the web server may send them to the correct web browser. This is essentially the same procedures that the Internet Map Server performs, but via CGI scripts that are written by parties other than ESRI. For ease of setup and maintenance, the first method is likely to be the more attractive of the two.

As one possible outcome of this study is the implementation of a similar suite of applications by Tri-Met, it is relevant to discuss pricing issues. Map Objects costs $1995 plus, either the cost of 10-seat license, if resale via commercial channels, or an in-house licensing charge. The license fees are $1000 and $2500 respectively. If used, Internet Map Server costs $500 for the program and $5000 per CPU to distribute maps over the Internet. It is assumed that a web server already exists, so this cost is not considered.

Conclusion

With the successful implementation of the first iteration as described above, the project has reached it first milestone: to show the feasibility of placing bus route maps on the Internet. While the current implementation can be considered crude, it does present the user with a useful tool in ascertaining where and when a bus will arrive at a time point. The next step is to refine the presentation of the data, and to make the maps more complete in regards to stops and cross streets, and to become more dynamic in terms of user interface and map creation.

References

[1] "Busview: A Regional APTS Experiment," D.J. Dailey, Transportation Research C, In preparation.
[2] http://www.its.washington.edu/projects/busview.html
[3] Advanced Public Transportation Systems, The State of the Art Update '96, U. S Department of Transportation, January 1996
[4] http://www.fta.dot.gov/library/technology/APTS/tws/travel11.htm
[5] http://www.smartraveler.com/

[6] http://www.itsmarta.com/Moreinfo.html
[7] Review and Assessment of En-Route Transit Information Systems. U.S. Department of Transportation, July 1995